Techniques for improving the hydraulic automatic simulation package (HASP)

Wang, Liming

Award date:
1988

Awarding institution:
University of Bath

TECHNIQUES FOR IMPROVING

THE HYDRAULIC AUTOMATIC SIMULATION PACKAGE (HASP)

submitted by

LIMING WANG

for the degree of Ph.D.

of the University of Bath

1988

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

SUMMARY

This thesis continues the improvement and development of the Hydraulic Automatic Simulation Package (HASP) by developing simulation techniques to avoid or to solve some of the common problems in the simulation of hydraulic systems, by modifying the program generator and writing component models.

Two kinds of model linking difficulties of simulation packages have been examined using HASP and CSMP. The solving approaches used an iteration loop for implicit algebraic loops, and a combination model or pseudo-state variable for linking problems between non-memory models. The model linking philosophy has shown that these approaches are valid for breaking linking obstacles.

An iterative technique has been applied to the modelling of components to avoid the mathematical stiffness problems. The iterative procedure in these models is a half interval technique. A two stage relief valve model and two pipe models with iterative procedure do not take into account the fluid compressibility in pilot chamber or in short pipe chamber, and their pressure values are computed instantaneously by iteration. A shaft model with an iterative procedure does not consider shaft rigidity and obtains its angular acceleration and torque by iteration, angular speed by integration. Simulation tests show that using these models with iterative procedure the mathematical stiffness problems can be avoided.

Single step numerical integrators are shown in the thesis to have advantages for coping with discontinuous and stiff systems compared with multistep methods. Two important modifications for standard Runge-Kutta-Merson and Runge-Kutta-Fehlberg algorithms have been effected in order to cope with discontinuity and stiffness problems. These two algorithms and a fixed time step fourth order explicit Runge-Kutta algorithm have been implemented in HASP. Simulation tests were carried out to demonstrate the operation of these integration algorithms and give comparisons of simulation results and speeds. The relevant computer modelling technique has also been simplified by using conditional statement method. It gives a simple way to code the hydraulic components described by discontinuous equations.

The simulation techniques used involve the modification of the HASP program generator and development of hydraulic component models. These are shown to have general purposes in the simulation of hydraulic systems using HASP.

ACKNOWLEDGEMENTS

The author would like to thank a number of people of the Fluid Power Centre for their assistance during the project.

The author wishes to express his sincere thanks to his supervisor, Professor D.E. Bowns for his invaluable help and guidance throughout the project. Without his assistance the work would never have been completed.

Thanks go to Drs. S.P. Tomlinson and D.G. Tilley for their help and the many useful discussions relating to the project, to Dr. K. Edge for his valuable advice.

The author also thanks the Chinese Government, the Fluid Power Centre and the Committee of Vice-Chancellors and Principals for funding the scholarship, studentship and the overseas research students award.

Finally, the author would like to thank his wife for her understanding and encouragement.

CONTENTS

CHAPTER 3 INVESTIGATION OF COMMON DIFFICULTIES IN SIMULATION OF HYDRAULIC SYSTEMS

CHAPTER 4 ITERATIVE TECHNIQUE FOR AVOIDING STIFFNESS PROBLEMS IN THE SIMULATION OF HYDRAULIC SYSTEMS

NOTATION

A	- Area
C	- Constant
C_d	- discharge coefficient of the orifice
C_s	- Slip coefficient
d	- Diameter
d_0	- Proportion of air dissolved in the fluid at STP
d_{sat}	- Proportion of air dissolved in the fluid at saturation pressure
D	- Volumetric displacement per radian Diameter (Chapter 4)
f	- Viscous friction coefficient Derivative of state variable (Chapter 6)
f_w	- Windage coefficient
F	- Force
F_f	- Friction force
F_s	- Stiction force
F_c	- Coulomb friction force
G	- Gear box ratio
h	- integration time step (Chapter 6)

I	- Polar moment of inertia
J	- Inertia
K	- Spring stiffness
k	- Constant
L	- Length
M	- Mass
n	- Polytropic index
P	- Pressure
P_{cr}	- Cracking pressure
P_{sat}	- Saturation pressure
ΔP	- Pressure drop
$\frac{dP}{dt}$	- Rate of change of pressure
Q	- Flow
S(t)	- Stiffness ratio
T	- Torque Time (in model coding)
T_s	- Stiction torque
T_c	- Coulomb friction torque
t	- Time
v	- Velocity

V	- Volume
y	- Displacement State variable (Chapter 6)
X,Y	- Displacement
W	- Power loss
α	- Angle of poppet valve
β	- Fluid bulk modulus
ϵ	- Error (Chapter 6)
ρ	- Density of the oil
λ	- Eigenvalue
μ	- Fluid absolute viscosity
ω	- Angular speed
$\dot{\omega}$	- Angular acceleration
θ	- Angle of inclination of actuator rod to horizontal
Δf	- Flow iterative error tolerance
Δp	- Pressure iterative error tolerance
ΔT	- Torque tolerance (Chapter 5)
$\Delta\dot{\omega}$	- Angular acceleration tolerance (Chapter 5)

Subscripts

D	- Driving
E	- Engine
e	- Effective
f	- Fluid
L	- Load
m	- Motor
p	- Pump Pipe (Chapters 4 and 6)
r	- Relief valve
s	- Shaft
1	- Relief valve main stage Input side of shaft (Chapter 5)
2	- Relief valve pilot stage Output side of shaft (Chapter 5)
3(c)	- Contraction cross section

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 This thesis is concerned with the techniques for improving the Hydraulic Automatic Simulation Package (HASP). In particular, it deals with techniques for avoiding model linking difficulties, iterative techniques for avoiding the mathematical stiffness in simulation, numerical integration techniques for solving stiff and discontinuous systems, and simplification of computer modelling techniques for discontinuous component models.

1.2 **Background of Digital Simulation Packages.** Digital computer simulations have the advantage that they can be run on a wide variety of computers and can cope with all kinds of nonlinearity. However, mathematical modelling and computer programming are time consuming and expensive procedures. In order that the engineer makes the use of digital simulation, a number of general purpose simulation languages and specialist application packages have been developed, for instance, CSMP (Continuous System Modelling Program)[1],[2] and [3], ACSL (Advanced Continuous Simulation Language)[4], CSSL (Continuous System Simulation Language), HASP (Hydraulic Automatic Simulation Package)[6],[7],[8] and [15], and DSH (Digital Simulation of Hydraulics)[9],[10].

1.3 **General purpose simulation languages.** It is recognised that the digital computer has the potential to be an important tool in the simulation of physical systems. It is also recognised that in order to utilise this tool to its full advantage, it is desirable that general purpose simulation languages be available. General purpose simulation languages allow the digital computer to be programmed in the same way as an analogue computer. For instance, CSMP provides elements which simulate simple integration, first and second order system behaviour and many non-

linearities. Not least, it provides for the inclusion of user written subroutines. General purpose simulation languages also take into account the inherent difficulties of programming and provide sorting procedures which allow the statements defining the dynamic or static behaviour of each component to be inserted into the program in any order. However, if these general purpose simulation packages are to be used as tools the user has to have programming skill. In general the problem with such simulation languages lies in their versatility. Although there are those that offer some form of block representation for simple mathematical elements and transfer functions, this facility is usually limited and it is common that the user is required to break down his physical systems, and in fact the user works close to the mathematical environment of the problem and has to be involved with considerable computer programming and simulation coding considerations. These requirements can only deter the engineer from making full use of this potentially useful design aid.

1.4 **Special purpose packages.** A number of specialist application packages have been developed to reduce further the need for specialist knowledge by concentrating on some particular types of physical systems. The versatility of such packages compared to general simulation languages is limited. However, within the discipline of some specialised field, it is a potent tool due to its direct application and its simplicity of use. For instance, several special purpose packages have been developed to simulate hydraulic systems which are described below respectively.

1.5 **Simulation Packages of Hydraulic Systems.** The Hydraulic Automatic Simulation Package (HASP) was developed at the Fluid Power Centre of Bath University for use by engineers, in which an automatic program generator was designed to avoid the user becoming involved with computer procedures. HASP is a package based on component models and the user of HASP is only required to build a computer block diagram by means of a circuit diagram of the hydraulic system being investigated so that the component data can be supplied to form a computer simulation program of the system. A full description of the HASP simulation package will be given in the subsequent sections.

1.6 Another special hydraulic simulation package based on the component models, which is termed DSH (Digital Simulation of Hydraulics), has been developed by the Aachen Industrial University in West Germany to simulate the performance of

hydraulic systems. A general structure of DSH is shown diagrammatically in figure 1.1. The package comprises eight main 'modulars' which were designed to respectively do the interpretation of input statements, generation of system models, simulation in time domain, analysis in frequency domain, linearisation of models, analysis of root locus, optimisation of parameters and graphics representation. The modulars are written in FORTRAN 4 language and linked to each other by interface programs. The integrator used is the explicit fourth order Runge-Kutta method. Comparing with the HASP simulation package, DSH package is much more complex and more difficult to use because the user is required to learn a high level language called DSH user language in order to supply the details of hydraulic circuit configuration in the form of data fields. Moreover the interrelation between components is established using a data file defined by special statements of user language rather than a source program. The simulation program including the system model, parameter input and result output are represented as complex data files. In this case, the user is still required to have a knowledge of the languages and coding consideration in the data file form. It also has been found inconvenient to input simulation data for components using specified data tables.

1.7 HYTRAN (Hydraulic Transient Analysis)[11] is one of a group of programs issued by the MacDonnel Douglas Aircraft Corporation. The other programs issued are SSFAN (Steady State Flow Analysis)[12], HYTTHA (Hydraulic Transient Thermal Analysis)[13] and HSFR (Hydraulic System Frequency Response). SSFAN calculates the steady state pressures and flows throughout a system under any loading conditions. An iterative procedure is used whereby flows are varied to obtain a pressure balance throughout the whole system. HSFR calculates the system frequency response using a form of the impedance method. HYTRAN was originally developed for the dynamic analysis of aircraft hydraulic systems. Since this program allows for distributed parameter models and uses the method of characteristics [20], the 'water hammer' equations and the partial differential equations describing wave propagation along pipe can be solved. Within HYTRAN the program SSFAN is used to set up initial conditions and the method of specified intervals is used with the method of characteristics and the solution requires both iteration and interpolation. The dynamic models of different systems can be constructed from a model subroutine library and are solved using an explicit fourth order Runge-Kutta integration algorithm. However the user is required to supply the coding to establish a simulation for a hydraulic circuit, and hence it is very difficult for the engineer to use or understand it.

1.8 **A Description of HASP Simulation Package.** The Hydraulic Automatic Simulation Package developed at the University of Bath is used to simulate the steady state and dynamic behaviour of hydraulic systems and components. The package is intended for use by engineers who do not have a high degree of computational skill or expertise in mathematical modelling. This package may be used at either the design stage or at system operation stage for analysing and predicting the behaviour of hydraulic systems and components. It has also been extended to account for the behaviour of mechanical loads and also associated electronic control circuits, and hence it has become possible to simulate the computer control of hydraulic systems, performance of mechanical-hydraulic systems, electronic-hydraulic systems or mechanical-hydraulic-electronic systems. The principle aims of the package are:

i) to provide a program which can be used by an engineer or scientist with little or no previous experience of computer systems and packages;

ii) to provide a program which allows changes in component parameters, or indeed in circuit configuration to be carried out without the need for programming by the user;

iii) to provide a library of models, each model representing the behaviour of a particular component of the general system being investigated.

1.9 **General structure of HASP.** is shown diagrammatically in figure 1.2. The package consists of three main parts below.

i) the program generator

ii) the component model library

iii) the numerical integrators

1.10 The centre of the package is the program generator. The main purpose of the program generator is to write a few FORTRAN control segments which link existing component models with the numerical integrator to form a complete simulation

program. The other two important tasks of the program generator are:

i) analyse the circuit data required as input and check for errors or model incompatibility.

ii) arrange the arbitrarily ordered models into an acceptable calling sequence of component subroutines in the system model segment AUX.

1.11 The HASP model library consists of over two hundred models that represent physical components including hydraulic, mechanical and electronic components. A special data file COMPON.DAT contains details of the important general features of models. The program generator reads the attribute details of each component model from this data file to check the validity of components of a system being investigated. The attributes include such information as link and signal details as well as the number of state variables used by a model.

1.12 At the commencement of this research work, there was one numerical integrator available in the package, i.e. GEAR. The Gear integrator is an implicit multi-step method. The order of this method can be automatically varied according to the stiffness of the problem to be solved[32]. The integration stepsize of this method is controlled by a mixed error test corresponding to the solution of state variables. This method is particularly suitable for the numerical solution of stiff systems, but is at a disadvantage when solving discontinuous systems. The current version of Gear integrator was developed by Caney[34] and Richards from work carried out by C.W. Gear[22]. A detailed description of Gear's method is given by Tomlinson[15].

1.13 Three numerical integrators have been added into HASP by the author and they are given below.

1. FOURTH ORDER EXPLICIT RUNGE-KUTTA
2. KUTTA-MERSON
3. KUTTA-FEHLBERG

These three integrators are explicit single step integration algorithms and they may be considered for solving the discontinuous systems[22],[23],[33]. However, fourth order explicit Runge-Kutta method is only suitable for the simulation of systems with less stiff problems because it has a fixed integration step. Kutta-Merson is one kind of explicit Runge-Kutta method with fourth order and variable stepsize. This method has higher computational efficiency than other Runge-Kutta methods with same order. It also can be used for solving stiff differential equations since two important modifications have been added into its computer coding. Firstly, a modification was added by Barker(1969), in which an error test determining logical variable is used to aid controlling integration stepsize. A second modification has been added by the author, in which a minimum integration stepsize h_{min} is specified to control the numerical error test. If the integration step is less than h_{min}, the error test for the estimation of state variables will be passed and the estimates of state variables are considered to be acceptable, and hence the possible infinite loop of error evaluation will be avoided. Similar with to Kutta-Merson, Kutta-Fehlberg also is a kind of Runge-Kutta method with fifth order and variable stepsize. This method has wider stability region and higher computational accuracy than Kutta-Merson, and hence its computational speed could be fast in many simulation cases. A detailed discussion of these integrators is given in Chapter 6 of this thesis.

1.14 The essentially continuous models and artificially continuous models which are smoothed by using cubic smoothing technique are suitable for all the HASP integrators. However, some recent models incorporating discontinuities which are modelled by using conditional statement technique can only be used with single step methods such as Runge-Kutta, Kutta-Merson or Kutta-Fehlberg methods. The conditional statement technique proposed by the author simplifies considerably the modelling of components with discontinuities and a description of this is given in the paragraph 3.19. The models using this technique are identified in the system by the letter R in the third position of the four character mnemonic name, eg. ALR0.

1.15 **Principle of using HASP.** In order to use the package, the user is initially required to construct a link block diagram representing the circuit under consideration. This diagram, developed by Bowns and Rolfe[21], is a representation of how individual models of hydraulic components are connected together in a circuit. For example a valve and actuator system shown in figure 1.3 is reduced in stages through the form shown in figure 1.4 to the link block diagram shown in figure 1.5. The user then converts the link block diagram (figure 1.5) into a table of

information in a format acceptable to the HASP program generator. The data corresponding to figure 1.5 is shown below.

```
07
DE0001  1
DE0101  2
DC0101  2  1  3  5  6
PI0601  3  4  1
AL0001  4  7  8  1  2
PI0602  7  6  2
TK0001  5
```

The first line indicates that there are seven component models in the circuit. The remaining lines list the component model names and define the links between them. In each line the four character is the mnemonic of component model, the two digit identifier indicates multiple occurrences of the same component model and other digits may be external, internal and signal links. For instance, in the fifth line AL00 is the name of a linear actuator model and 01 shows this model is used once in the circuit. The pressures in both piston and rod chambers are on the external links 4 and 7, the displacement and velocity of actuator are on the internal link 8. The variable volumes in both chambers are on the signal links 1 and 2 respectively. This data forms the input required by the program generator.

1.16 Subsequently, the program generator checks the validity of this input data file and then sorts a correct calling sequence of component models for the simulation by using the sorting subroutine PGCOMP of HASP. The program generator then writes the control segments of the simulation program in FORTRAN 77 language by the HASP PGOUT subroutine. The constituent parts of the simulation program CAD.EXE and the model selector file written by the program generator are:

1) **Main segment MAIN.** This calls the segments which are (i) used to input the dimensional and performance data details and then (ii) call the integration subroutine necessary to perform the simulation.

2) **Parameter input segment CONTRL.** This controls the input of

simulation parameters, time step, and it is called from the main segment MAIN.

3) **System model segment AUX.** This contains the calls to the component model calculation subroutines in a order which is specified by a sorting algorithm, and it is called by the integration and result output subroutines.

4) **Output segment OUT.** This stores the simulation results in a data file CADRES.DAT necessary for either a graphical display or as a printout.

5) **Component model selector file CAD.OPT.** This selects the required component model subroutines from the library.

The relevant component models are selected from the library in terms of CAD.OPT file and added to the system model segment. Finally, the simulation program called CAD.EXE is formed by linking the integrator chosen and component models to these control segments. The simulation program may then be run, and to continue the process shown in figure 1.2, ie. the input of simulation parameters, the performance simulation of systems and output of simulation results.

1.17 **Difficulties in Simulation and Choice of Calculation Methods.** There are three types of common difficulties in the simulation of hydraulic systems. These are:

1) The problem of linking component models.

2) The problem of physical discontinuities.

3) The problem of system stiffness.

1.18 **Linking difficulty.** Two types of linking difficulties might appear at the program generation stage. The first is caused by an implicit algebraic loop, and the second is caused by the linking a certain class of component models. These might

affect the generation of a correct system model and even make a simulation abort at the program generation stage. A full description of model linking difficulties and relevant solving approaches, and the general philosophy of linking computer models are given in detail in Chapter 2 of this thesis.

1.19 **Stiffness difficulty.** The simulation of a hydraulic system involves the formulation and integration of a set of differential equations describing its dynamic response. If the solutions of differential equations have widely differing decay rates[22][23][32], the system is said to be stiff. The problem of stiffness in the simulation of hydraulic systems is usually caused by pressure transients in pipes, mechanical end stops in actuators and thermal transient effects. A typical example often met with is the simulation of a system involving a restriction linking two pipes. This has been dealt with by Bowns and Rolfe[30]. In this thesis it is proposed to avoid the stiffness problem of hydraulic systems by using a half interval iterative technique, the basic principle of which is to instantaneously compute the pressure in a pipe by iteration rather than by integration. The similar technique also has been applied to avoid the stiffness problem caused by the mechanical shaft in the shaft coupled power transmission systems. These techniques are discussed in detail in the Chapters 4 and 5.

1.20 In terms of computational efficiency, the best numerical integration methods for solving stiff problems are linear multistep methods[15]. Gear's method is a widely used multistep method currently available for solving stiff problems and has been implemented in HASP. The routine automatically choses the stepsize and the order of the method in an attempt to obtain a specified accuracy with a minimum of computation. It is also recognised that some variable step length Runge-Kutta methods can be successfully used for solving stiff systems and two of them have been implemented in HASP by the author. A discussion of these algorithms are given in the Chapter 6 of this thesis.

1.21 **Discontinuity difficulty.** Although Gear's method is capable of coping with considerable stiffness, it is prone to fail when it reaches a point of discontinuity because the current version of Gear's routine (in HASP) is only suitable for continuous models. Two types of discontinuity could make current multistep integration methods fail. These are[76],[78]:

i) discontinuities where a dependent variable jumps instantaneously from one value to another, and

ii) discontinuities where the time derivative of a dependent variable jumps instantaneously.

An example of types i) and ii) could be an actuator or valve hitting a stop. When an actuator or valve hits an end stop, both types of discontinuities are met with. If the state variables are displacement and velocity, velocity is of type i) and the derivative of velocity is type ii). In order to solve this kind of problem, a cubic smoothing technique has been described by Tomlinson[15] to smooth the discontinuities of models over an extremely small region.

1.22 Two problems with the use of cubic smoothing technique are the possible inaccuracy of numerical solutions and the excessive coding of component models. In the model of an actuator AL00, for instance, hypothetical end-stop spring stiffness and damping forces are required when smoothing the discontinuities at the end-stop. Since the real end-stop reaction force acting on the piston is unpredicted, unreasonable simulation results could be obtained when these forces specified by the user are used. Figure 1.6 shows incorrect numerical solutions of actuator displacement and velocity responses at the beginning of the end-stop. Moreover the coding describing smoothing regions has to be complex, and it is difficult to understand.

1.23 Runge-Kutta methods mentioned earlier are specially suitable for discontinuous systems because they are single step methods[22],[23],[33]. Therefore, the discontinuities of component models can be directly modelled by using conditional statement technique. A discussion of this is given in detailed in paragraph 3.19.

1.24 **Plan and Scope of This Thesis.** This thesis is divided into seven chapters.

Chapter 1 is the introduction of the thesis.

Chapter 2 describes the general philosophy for linking computer models. It commences by introducing the general linking difficulties of computer models in general purpose and specialist application simulation packages, such as the linking difficulties concerning implicit algebraic loop, and between some algebraic component models. It then describes how to solve these difficulties which could happen at the simulation program generation stage. Several approaches of solving are proposed by the author, which comprise the combination model method, state variable method and pseudo-state variable method. The advantages and disadvantages of using these methods are also discussed. Model linking philosophy for system simulation is then described in detail using HASP (Hydraulic Automatic Simulation Package) as example. Finally a considerable attention is given to describe the sorting process of calling sequence of component models for HASP. A simple hydrostatic transmission system is used to validate the above. The relationship between the sorting algorithm in the HASP program generator and a data file termed as COMPON.DAT containing information defining the structures of the various models in the system is discussed in detail.

Chapter 3 gives the analysis of common difficulties in the digital simulation of hydraulic systems. The concept of mathematical stiffness and the stiffness difficulties in the simulation of hydraulic systems is discussed in detail by analysing simple hydraulic linear actuator circuit. Techniques for coping with mathematical stiffness are also proposed. Two definitions of the mathematical and physical discontinuity problems in simulation are put forward by the author. Techniques for coping with discontinuity difficulties, including cubic smoothing and conditional statement techniques, are also analysed.

Chapter 4 describes an iterative technique for avoiding the stiffness problems in the simulation of hydraulic systems. It commences by describing why and how the mathematical stiffness problems in the simulation of hydraulic systems can be avoided by using iterative techniques and then introduces a half-interval iterative

technique to model a two stage relief valve. It also describes how this iterative technique has been used to compute instantaneously the pressure in a frictionless pipe with a small volume and pressure in a linear actuator approaching the end-stop, and how the model linking problems encounted were solved. A description is given in detail for the iterative processes of instantaneous pressure values in pilot stage of relief valve, pipe and actuator. How the HASP program generator writes these iterative procedure subroutines automatically by means of different systems being investigated are described. Several simulation tests for component models with iterative procedure are given to check the reliability of models.

Chapter 5 describes the iterative technique for avoiding the stiffness problems in the simulation of shaft coupled power transmission systems. It commences with a description of how the stiffness problems caused by mechanical shafts in a power transmission system can be avoided and why the individual shaft model with iterative procedure is necessary to compute instantaneously the torque and angular acceleration on the shaft. An iterative technique is used satisfactorily to model an individual circular shaft and a shaft-gearbox unit. A typical simulation example of an engine-hydrostatic transmission system using shaft models with iterative procedure is given in detail.

Chapter 6 describes some single step integration methods for discontinuous and stiff systems. It commences by analysing the characteristics, accuracy and stable regions of seven typical single step numerical integration methods and describes why the single step variable time step methods, such as modified fourth order Runge-Kutta-Merson and fifth order Runge-Kutta-Fehlberg methods, may be able to cope with discontinuous and stiff systems. The philosophy for coping with stiff and discontinuous problems is described in detail. The description includes the operation of Gear integrator and computer modelling method using cubic smoothing technique for the components with mathematical discontinuity. It also stresses the operation of a single step Runge-Kutta-Merson method and the relevant modelling method of solving discontinuous problems by employing conditional statement technique. Two important modifications have been done in standard Runge-Kutta-Merson and Runge-Kutta-Fehlberg methods for solving extremely stiff and discontinuous problems. A description of the conditional statement (IF STATEMENT) technique employed to model discontinuities is given in detail. This chapter continues by describing why in the AUX subroutine, the reconversion of state variables and linking variables of a component model is necessary to ensure

that the discontinuous model with IF STATEMENTS can be solved correctly. It further describes the necessary information required to input into the COMPON.DAT data file by the user and how the program generator verify this information and writes out reconversion statements required in AUX. This chapter then describes the implementations of three modified single step integration algorithms for HASP. They are the explicit fourth order Runge-Kutta method, fourth order Runge-Kutta-Merson method and fifth order Runge-Kutta-Fehlberg method. The comparison tests of simulation for different HASP integrators (GEAR, RK4, KUTMER and KUTFEH) and different component models which were written by either cubic smoothing or conditional statement technique are finally listed.

Chapter 7 gives conclusions of this thesis and recommendations for future work.

Appendix A describes the modifications of the HASP program generator corresponding to the techniques developed by the author for improving HASP. This section commences by describing the structure of the HASP program generator and model attribute file COMPON.DAT related with program generation. This section follows by describing a whole process in which the program generator writes out the necessary iterative procedure subroutines for pipe model PIA5 or PIA6 automatically. This appendix concludes with a full description for the modification of the program generator corresponding to new numerical integrators.

Appendix B describes five computer models with iterative procedure of hydraulic components in the form of documentation reports which were originally written for the HASP model library at the University of Bath.

Appendix C describes ten computer models of hydraulic components in which the discontinuous mathematical equations were modelled by using the conditional statement (IF STATEMENTS) technique.

Appendix D lists the codings of three modified single step algorithms, which include explicit fourth order Runge-Kutta, fourth order variable time step Runge-Kutta-Merson and fifth order variable time step Runge-Kutta-Fehlberg methods developed for use with HASP.

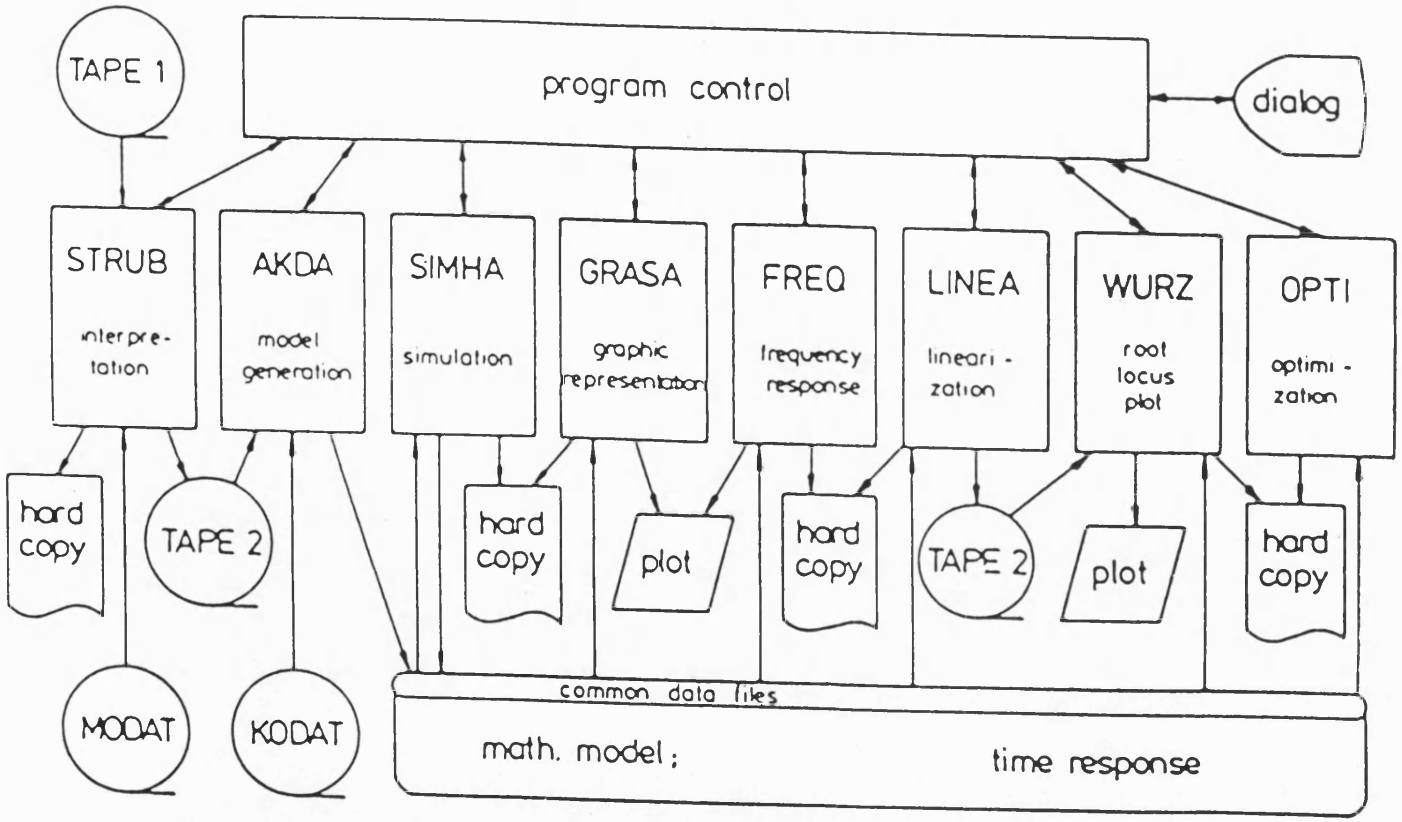


Figure 1.1 DSH - Program system

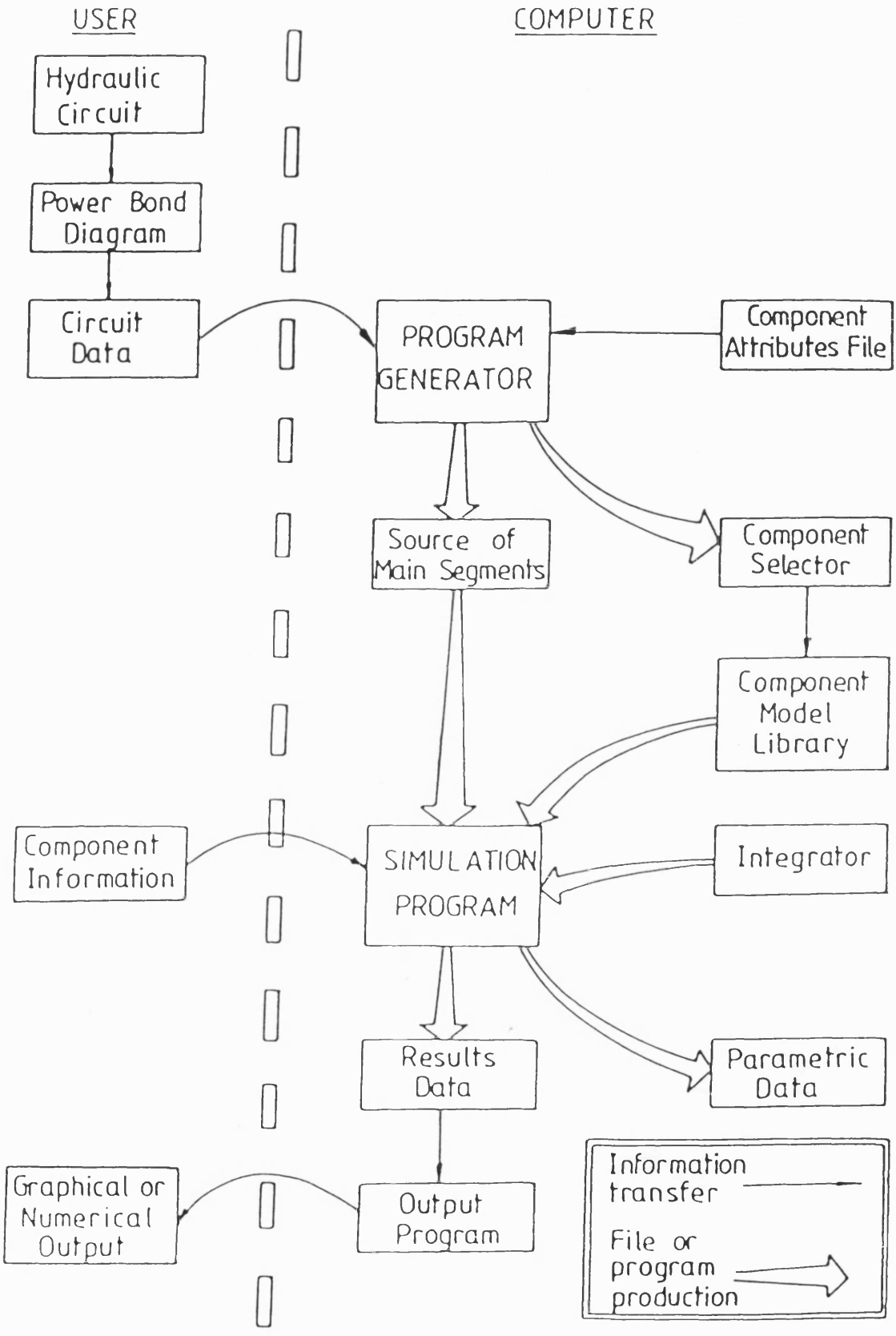


Figure 1.2 The structure of HASP

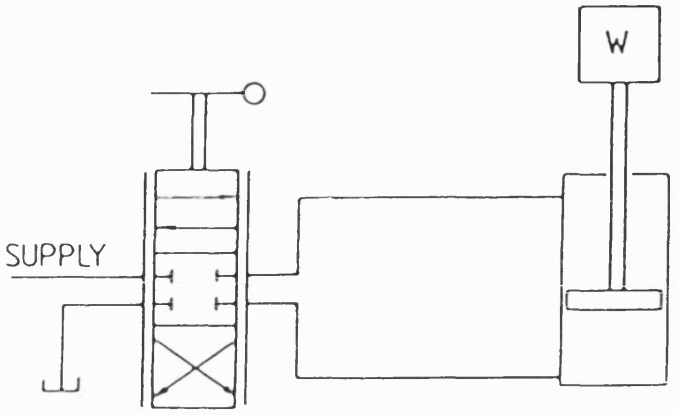
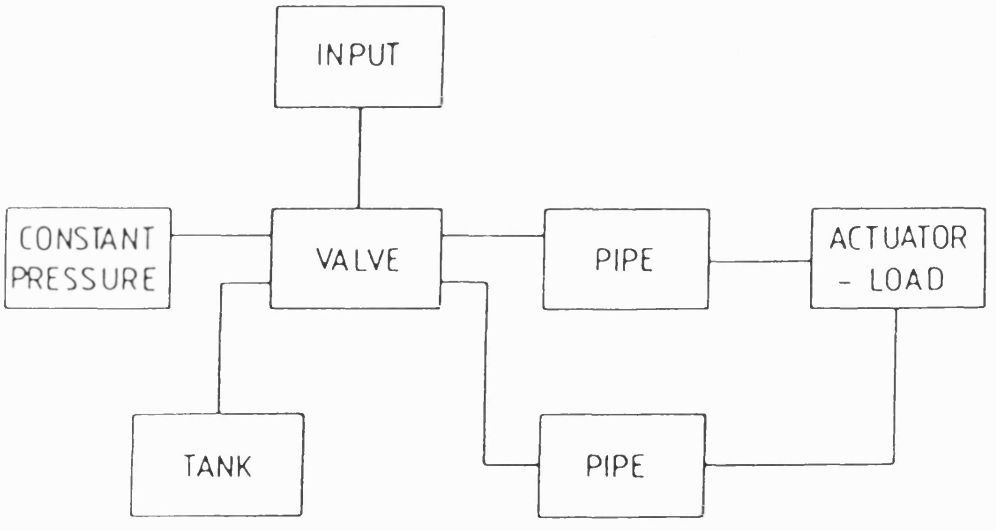


Figure 1.3 Valve and vertically mounted actuator



Figrue 1.4 Block representation of actuator system

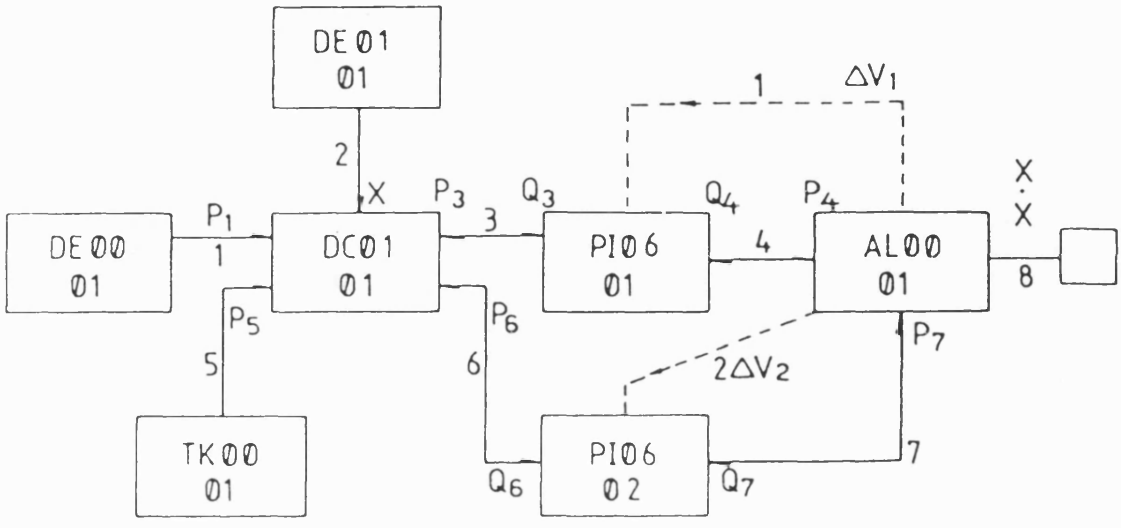
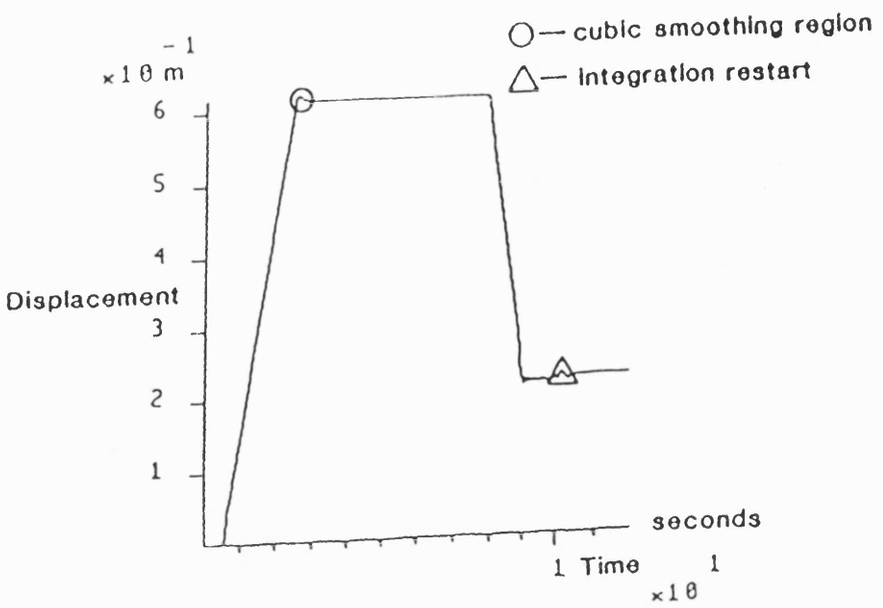


Figure 1.5 Power bond representation of actuator system



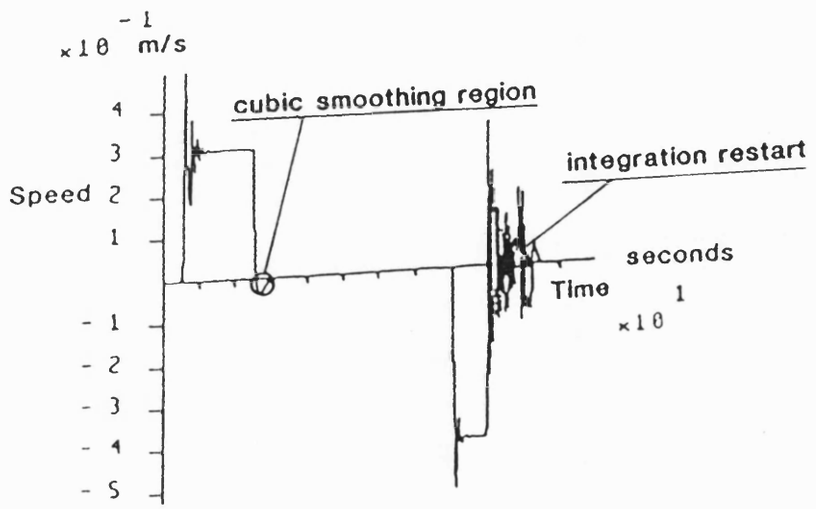


Figure 1.6 Displacement and velocity responses of actuator

CHAPTER 2

GENERAL PHILOSOPHY FOR LINKING COMPUTER MODELS

INTRODUCTION

2.1 At present, a number of general purpose simulation languages and specialist application packages, for instance, CSMP, ACSL, CSSL, HASP and DSH[1],[2],[3],[4],[6],[7],[8],[9],[10] are available for evaluating the dynamic or steady state performance of engineering systems. In these languages or packages, the fundamental principle of computer model linking in the system simulation is identical although the system model structures and modelling techniques adopted may be different. However, the modes of system model linking might affect the validity of computer system modelling and even the generation of the simulation program. In this chapter, the Continuous Simulation Modelling Program (CSMP) and Hydraulic Automatic Simulation Package (HASP) will be used to illustrate the philosophy of linking computer models and linking difficulties. Several approaches for solving linking difficulties encountered in system simulation will be proposed.

MODEL LINKING DIFFICULTIES

2.2 There are two kinds of linking difficulties might affect the generation of the system model to be simulated. The first difficulty for the model linking is caused by an implicit algebraic loop, and the second one is caused by linking some component models. The former is a general problem in many simulation languages or packages and the latter is a special linking problem in the HASP simulation package in which standard component model subroutines are separately put in its model library. The

crux of these two kinds of linking problems is that the solution of the loop model or component model cannot be obtained due to or lack of input information of the model, such as initial conditions or previous computed values, or there is an implicit relationship between models.

Linking Problems about Implicit Algebraic Loops

2.3 A general implicit algebraic loop is defined by a set of algebraic equations or models containing a non-memory function or model. The term "memory" and "non-memory" elements are mentioned in Ref[1] but not defined. In the following work the following definitions have been adopted.

(i) **Memory elements.** These are functions or models which can retain and supply initial values, or from which output values are obtained by integrator. For instance, the functions or models which imply integration, time delay functions and input functions are all memory models.

(ii) **Non-memory elements.** These are those functions or models whose solutions depend on inputs supplied by other functions or models. Examples are algebraic functions or models, and also models which supply algebraic variables to other models.

2.4 As a simple example of the analysis, consider an algebraic loop defined by the following single statement:

$$Y = ALGE^{*}(X-Y) \qquad \ldots\ldots (2.1)$$

where the function ALGE could be an algebraic equation or a constant. Y appears both as an input and output and the relationship is implicit, and this would still be

true if the equation were to be written as several equations, thus

$$\begin{aligned} A &= ALGE*X \\ B &= ALGE*Y \\ Y &= A - B \end{aligned} \qquad \ldots\ldots (2.2)$$

The last two equations constitute an implicit algebraic loop and it will now be shown that the sorting algorithm of any simulation language or package could not process this set of equations. Of course, if the loop model described by equation (2.1) could be considered as a single model in general simulation languages or packages, the implicit relationship would be removed by rearranging the mathematical equation of the loop model as follows:

$$Y = \frac{ALGE}{1 + ALGE} X \qquad \ldots\ldots (2.3)$$

In this equation, if ALGE is a known constant, the variable Y is a function of known values X and ALGE and can be directly solved. In this case, the previous implicit algebraic loop disappears and an explicit model described in equation (2.3) is formed. However it is often extremely inconvenient to write the equation in this form, particularly if the function ALGE is involved, and requires much effect in modelling.

2.5 The implicit algebraic loop described in equation (2.1) is shown in figure 2.1. Since the variable Y is an unknown value and the algebraic model ALGE in this loop is a non-memory one, the numerical solution Y of this loop cannot be directly obtained in terms of the unknown value Y itself and another known input X. In other words, in this kind of implicit algebraic loop, there is a linking difficulty between two non-memory functions or models. Figure 2.2 shows the system in block diagram form ALGE representing one function and SUM, an element combining X and Y.

2.6 **Proposed Solution Methods for Implicit Loops.** The same difficulty will occur in other kinds of implicit algebraic loops, such as an implicit loop with an algebraic feedback and an implicit loop with an algebraic forward function and an algebraic feedback which are shown respectively in figure 2.3 and figure 2.4. In fact, the essence of an implicit algebraic loop is that it does not contain memory functions or models. If it can be modified to a memory loop, then the implicit relationship between functions or models could be broken. For instance, implicit algebraic loops which cause the linking problem between two models or functions could be broken by using an integration function or model to replace ALGE, or by using an iterative technique to obtain the approach solution of an implicit loop.

2.7 **Integration loop method.** If an integration function INTE replaces the function ALGE, the previous algebraic loop shown in figure 2.1 becomes a memory loop with an integration as shown in figure 2.5, and the equations describing this loop are written as following:

$$\text{SUM} = X - Y$$

$$Y = \frac{K}{S} SUM$$

$$= \frac{K}{S}(X - Y) \qquad \ldots\ldots (2.4)$$

where S is the Laplace operator

Although Y also appears both as an input X and output Y, the previous implicit relationship of two sides of equation has been broken by the integration function shown in a transfer function form $\frac{1}{S}$. The Y variable on the left side of the equation is an unknown value to be computed at current time point t, and Y on the right side of same equation is a known value which is either an initial value or a previously computed value at last time point $t - dt$. Since the input X also is a known value, the relationship shown in equation (2.4) is explicit, and hence the Y variable at the time point t can be solved without any difficulty. In fact, the equation (2.4) can be written down in a differential equation as follows:

$$Y = K \int (ERROR) dt + IC$$
$$= K \int (X - Y) dt + IC \qquad \ldots\ldots (2.5)$$

The replacement of ALGE by an integrator can only be done by ensuring rapid integration. This can of course lead to the use of very short time steps and corresponding large computation times.

2.8 **Iterative loop method of CSMP.** The CSMP library supplies an implicit iterative function IMPL for breaking an implicit algebraic loop, which is shown in figure 2.6. The use of the iterative function IMPL causes a subiteration to be performed for each increment of the independent variable until the algebraic relationship within the loop is satisfied[1]. In that sense, the implicit algebraic loop becomes the memory loop.

In order to use the implicit function IMPL, the user first writes a structure statement of the form

$$Y = IMPL (IC, P, FOFY) \qquad \ldots\ldots (2.6)$$

where Y is the variable whose convergence is to be tested, IMPL is the name of the implicit functional block, IC is the initial guess provided by the user, P is the error criterion to be met and FOFY is the output name of the last statement in the definition of the implicit algebraic loop. The IMPL statement must be followed immediately by the statements evaluating FOFY. As an example of using this function, consider a same implicit algebraic loop shown in equation (2.2) which is rewritten as follows:

$$A = ALGE \times X$$

$$B = ALGE \times Y$$

$$Y = A - B$$

The above implicit relationship between statements B and Y could be solved by using the implicit function IMPL described above, and a relevant loop model segment is arranged below.

```
Y = IMPL(Y0,ERROR,FOFY)
A = ALGE*X
FOFY = A - ALGE*Y                                    ......(2.7)
```

In this case, the implicit algebraic loop is broken and the output variable Y could be obtained by iteration.

2.9 **Iterative loop method suggested for HASP.** In the HASP simulation package, the author has suggested an iterative model IMIT to break down the implicit algebraic loop discussed above in a similar manner. The iterative algorithm of the model IMIT is given below.

As an example of the discussion, consider an algebraic loop with the iterative model IMIT which shown in figure 2.7. The known input of the loop is X and the output is Y. The model ALGE could be an algebraic equation or a constant, and the model SUM is a summing algebraic equation which can be expressed by:

```
SUM = X - Y
```

The model IMIT consists of a standard iterative formula with an error criterion specified by the user, and an estimation equation of the iterative searching point ZGUESS. The operation process of searching a correct output value Y at time point t is given by:

2.10 **First step: set initial Z value.** In the computation of this algebraic loop, the iterative model IMIT is defined as a memory model and its initial output value is set as ZGUESS by the user.

2.11 **Second step: iterative loop of correct Z value.** After obtaining an initial ZGUESS, a correct output Z of the model IMIT can be evaluated due to the following procedure:

(a) estimated output value YGUESS of the loop

YGUESS = ALGE * ZGUESS

(b) error estimated value ZI of the loop

SUM = X - YGUESS
ZI = SUM

$$\Delta Z = ZI - ZGUESS$$

(c) error analysis

$$| ZI - ZGUESS | \leqslant \epsilon \qquad \ldots\ldots (2.8)$$

If it is true, the correct Z value is obtained and is given by

Z = ZGUESS

and then an improved solution is obtained by putting:

Y = Z * ALGE

If the error criterion is not met, a new iterative searching point is set by:

$$ZGUESS = \frac{ZGUESS + ZI}{2} \qquad \ldots\ldots (2.9)$$

and then the iterative computation is repeated as the operation goes back to (b) until the error criterion provided by the user is satisfied. A flowchart of this iterative algorithm for breaking an implicit algebraic loop is shown in figure 2.8.

Linking Problems between Non-memory Component Models.

2.12 As an example of analysis, consider a simple feedback control system as shown in figure 2.9. This system consists of a proportional feedback block and a forward control block that manipulates a proportional control process. Assuming that the system is expressed by four separate component functions or models called CTRL, ALG1, ALG2 and SUM1. They are given by:

(a) for forward controller CTRL:

$$\text{MANIP} = \int \text{ERROR} + \text{IC}$$

(b) for forward proportional controller ALG1:

OUTPUT = K1 * MANIP

(c) for feedback proportional controller ALG2:

FEEDBACK = K2 * OUTPUT

(d) for control error SUM1:

ERROR = INPUT - FEEDBACK

Here the CTRL is a memory model which has an initial condition or previous computed value from the integrator and the ALG1, ALG2 and SUM1 are non-memory models. The block diagram of the system is shown in figure 2.10.

2.13 **In a General Simulation Language Such As CSMP** , the model linking is represented by the block diagram as shown in figure 2.10. Since information transfer in this block diagram is unidirectional, the linking problem between non-memory component models can be avoided if the computation sequence of system is correct. For instance, the calculation of the values at the time point t can be carried out by chosing the following correct computational sequence.

$$Y(t) = AGL1 * Z(t)$$
$$F(t) = AGL2 * Z(t)$$
$$E(t) = X(t) - F(t)$$
$$Z(t+dt) = \int E(t)\, dt$$

This computational sequence is arranged by the sorting algorithm of CSMP.

2.14 **In HASP.** HASP is a component based language, the component models, integrator are written separately as subroutines. If the model linking of a system were carried out using the above block diagram with a single direction, the linking problem between two non-memory models such as ALG1 and ALG2 would be avoided. However, the model linking of the system in HASP is generally carried out by a power bond diagram with dual direction[7],[8],[89]. Figure 2.11 shows such a HASP power bond diagram of the control system discussed above. In this system, the calculation of each component model requires known input values from adjacent models to which it is connected. This is true in spite of the fact that in the example the model ALG1 does not use the input F and the model ALG2 does not use the input E. Since non-memory models can not supply output values without known input values, the relationship of input and output information between each other is implicit, and hence the connections between models ALG1 and ALG2, and between models ALG2 and SUM1 will be impossible. In this case, the diagnostic procedure of the HASP program generator gives a warning message "IMPLICIT RELATIONSHIP BETWEEN MODELS".

Therefore, in a HASP simulation system, every non-memory component model must be linked with adjacent memory models such as CTRL which can supply known input values. Otherwise, the linking difficulty between non-memory component models will occur at the system simulation stage. For instance, in the system above, only models CTRL and ALG1 or SUM1 and CTRL can be connected without any linking problem.

2.15 In fact, linking difficulties between non-memory component models always exist at the HASP simulation stage. Therefore, it is very important for the user to choose correct models to avoid this kind of linking problem. In the HASP model library, except the memory models with state variables, there are some instantaneous memory models such as hydraulic tank models, electrical motor

models and duty cycle models. They have one feature similar to a memory component model with a state variable, i.e. the output ports of models have known values which can be supplied to adjacent models after the user defines the input data of models[8].

2.16 Approaches to Solution of the Linking Difficulty between Non-memory Models. The crux of this type of linking problem is that two or more non-memory component models require input information from the each other. Since non-memory models cannot supply output values without known input values, the relationship of input and output information between each other is implicit and it is impossible to determine which should be called first during the arrangement of a correct calling sequence. For this reason, if a system being investigated is being sorted by the sorting program PGCOMP of the HASP program generator, the diagnostic procedure of PGCOMP will give a diagnostic message "IMPLICIT RELATIONSHIP BETWEEN MODELS" and then stop the sort process. For instance, in a simple open loop hydrostatic transmission shown in figure 2.12, consider a hypothetical instantaneous rotary load model LR01. This model is a non-memory component model and its link diagram is shown in figure 2.13. It provides angular speed ω in return for torque T. If this model was connected to the hydraulic motor model MO00 which provides torque from pressure and speed information, the link diagram would appear as in figure 2.14. Since both T and ω are not state variables, LR0101 cannot be called before MO0001 because it requires T as an input which is given by MO0001. Similarly MO0001 also cannot be called before LR0101 because it requires ω as an input. In this case, an implicit relationship between two models occurs. There are three ways of solving this kind of problem.

2.17 Combination model. The rotary load model and hydraulic motor model can be combined into one and the algebraic equations for both can be solved simultaneously in the resulting combination model. Many other combinations of models are possible such as pump, prime mover and connecting pipe or pipe and relief valve. This is satisfactory in some cases but since each model in itself may be complex, the number of combinations would have to be great and modelling problems could be large.

2.18 **State variable method.** The rotary load model can be made a dynamic memory component with ω as a state variable which is on the external link of the model, i.e. using the dynamic model LR00 replaces instantaneous model LR01 in order to avoid the implicit relationship between two models MO00 and LR01.

2.19 **Pseudo-state variable approach.** The instantaneous load model gives algebraic equations for determining the angular velocity when a torque value is supplied from the algebraic motor model. If however a further statement

$$\frac{d\omega}{dt} = 0$$

is inserted in the model, ω becomes a state variable, but will take up the value determined by the algebraic equations. For instance, this load model is assumed to be:

$$T_m = K\ \omega^2$$

then the angular velocity is determined by:

$$\omega(t) = \sqrt{\frac{T_m}{K}} \quad \text{with} \quad \frac{d\omega}{dt} = 0,$$

the angular velocity supplied by the integrator is:

$$\omega(t+1) = \int \frac{d\omega}{dt}\, dt + \omega(t) = \omega(t) \qquad \ldots\ldots (2.10)$$

It should be seen that setting $\frac{d\omega}{dt}=0$ in the load model is simply a device to make it possible for the integrator to deal with ω as a state variable. This is necessary to ensure the correct linking and ordering of the models.

Obviously the ω value is calculated by the algebraic equation in the motor model rather than supplied by the integrator, and hence it is not a real state variable and only can be defined as a pseudo-state variable. In this case, however, the instantaneous model LR01 becomes a memory component model and can be linked with the non-memory instantaneous hydraulic motor model MO00 without any linking difficulty. This method has the disadvantage that it increases the number of state variables in a simulation.

This technique has been used in Chapter 4 in developing pipe model PIA5.

MODEL LINKING PHILOSOPHY

2.20 **Introduction.** In HASP, the models of a system to be simulated are first represented by a power bond block diagram and the relevant information is put in a circuit data file. The HASP sorting algorithm will then sort a correct calling sequence of the system model by means of the "model details" in circuit data file and information in model attributes file COMPON.DAT. This will be described in paragraphs 2.30 to 2.39. Finally the component models are linking in the correct calling sequence in the subroutine AUX.FOR. In this section, the model linking philosophy for system simulation using HASP and the relevant sort process of simulation systems will be discussed.

2.21 **Hydraulic Component Models in HASP.** In current HASP model library, most of component models are developed separately. The model of a large system can be obtained by linking the component models chosen in a system in terms of the HASP sorting algorithm. Each component is represented by a box in which a four character name of the component is written, and can be connected with other components by lines which have been termed "external links" and "signal links" described in Ref[15]. The physical inputs and outputs to each box are shown on links. For instance, a relief valve model is expressed by four characters PC01 and receives input pressures as effort variables from another components on two external links and outputs the flow rate as a flow variable through the valve on same links. The half-arrows corresponding to flow imply output of information rather than algebraic direction of flow. The links are numbered in a manner convenient to the user and a typical model for this valve is shown in figure 2.15. The component model subroutines used in HASP are written by modellers using standard coding and every component model is described by two separate subroutines, an input subroutine and a calculation subroutine. A detailed description about the structure of a component model is given by Tomlinson[15].

2.22 **Computational Sequence of Mathematical Equations in a Component Model.** Since all the hydraulic component models are written separately as subroutines, the computational sequence of the mathematical equations for each component model is very important for correctly simulating the performance of a component. A correct sequence ensures that the output of each equation in the component model is computed on the basis of equation input values for time t. An incorrect sequence would update an equation's output for time t with input values for time $t - dt$. This incorrect sequence would affect the validity of the solution. For instance, consider the mathematical model of a hydraulic component, which is a first order differential equation containing two algebraic equations ALGE and FUN. Its mathematical equations are assumed as follows:

$$ALGE = Y \times CON1$$

$$FUN = ALGE + CON2$$

$$\frac{dY}{dt} = ALGE + FUN \qquad \ldots\ldots (2.11)$$

where Y is a state variable and the $\frac{dY}{dt}$ is the derivative of Y, and the CON1 and CON2 could be algebraic functions or constants. If a computational sequence of above mathematical equations is arranged below by the model writer,

$$\frac{dY}{dt} = ALGE + FUN$$

$$ALGE = Y \times CON1$$

$$FUN = ALGE + CON2 \qquad \ldots\ldots (2.12)$$

the computation of the mathematical equations described would introduce a time delay for the solution and even makes some computational mistakes because the computation of $\frac{dY}{dt}$ does not have the knowledge of ALGE and FUN at time point

t. Therefore, a correct computational sequence for each component model is required and the relevant requirements are proposed below.

Computational Sequence Requirements for Model Writers

2.23 A correct computational sequence of the mathematical equations within each component model is one, in which the output of each equation statement can be computed using known input values for current time point t. Since the general mathematical models of hydraulic components may consist of algebraic equations, differential equations, or a combination of both types of equations, the computational sequence rules of mathematical equations for each component model should include:

2.24 **First stage.** Write down all the algebraic equation statements that contribute to the input of the differential equations. A correct computational sequence for these algebraic equations is determined in terms of the requirements between algebraic equations, i.e. in the HASP model calculation subroutine, the algebraic equation statement with known input values should be put before those statements with unknown input. Of course, the output of the algebraic equation statements with known input values would be the known input for subsequent statements. For example, consider the example under consideration, the algebraic equations will be put in the calculation subroutine in the following correct computational sequence:

$$ALGE = Y \times CON1$$

$$FUN = ALGE + CON2$$

where Y is known because it is a state variable, and hence the output ALGE of the first equation statement is known for the input of the second equation statement. For some statements which have known input values at the same time, they can be put in an arbitrary order.

2.25 **Second stage.** Determine the sequence for all differential equations in an arbitrary order. For the above example, the differential equation of the component model is:

$$\frac{dY}{dt} = ALGE + FUN$$

Eventually, a correct computational sequence of this component model is given by:

$$ALGE = Y \times CON1$$

$$FUN = ALGE + CON2$$

$$\frac{dY}{dt} = ALGE + FUN$$

here the derivative computed $\frac{dY}{dt}$ at current time point t will be sent to the integrator, and then the state variable value Y at next time point $t + dt$ will be provided by the integrator.

Obviously, in order to correctly simulate the performance of a hydraulic component, the correct computational sequence of mathematical equations in this component model must be arranged by the HASP model writer by means of above requirements.

System Model Linking

2.26 In order to form a system model, the component models of the system have to be linked together using a power bond block diagram. As an example, consider the simple open loop hydrostatic transmission shown in figure 2.12, which consists of a constant speed electric motor, hydrostatic pump, pipe, motor, rotary load and

relief valve. A block diagram, representing all component models necessary to perform this system behaviour, is shown in figure 2.16. From this diagram a circuit data file is obtained in order to perform the program generation. Here a linking details of component models for the system is listed below.

09
TK0101 01
PU0001 01 02 03
PM0101 03
PI0501 02 04 05
PC0101 04 06
TK0102 06
MO0001 05 07 08
TK0103 07
LR0001 08

The first line indicates that there are nine component models in the circuit being investigated. The remaining lines list the component models in any order and define the links between models. Each line includes the component model name specified by four characters, the two digit identifier to identify multiple occurrences of the identical model, and the links expressed by the two digit numbers separated by single blank spaces.

2.27 In model details listed above, details of the linking for each component model may be obtained by reference to documentation describing component models[8], and the numbering of links for whole system can be determined by the user in terms of the correct input-output relationship of each component model. In addition, it is necessary to specify when the same model is used more than once in a given circuit. For instance, a tank model is used for three times in this system, hence, appropriate model expressions in circuit data file have to be written respectively by:

TK0101 01
TK0102 06
TK0103 07

2.28 After supplying the "model details" of the circuit, all component models of this system seem to be linked together due to given linking numbers. However, a successful linking between models depends on whether the input-output relationship of models are matched or not. For example, in order to avoid the linking problems discussed earlier, an non-memory algebraic model has to be linked with one or more memory component models, otherwise the direct linking between models will be not matched. In addition, the transmission variables between models being linked must be identical. Model A links with model B, for example, the input of model A is effort variable and the output is flow variable, thus the input of model B must be flow variable and output must be effort variable[15],[89]. The check for these is done by an automatic diagnostic procedure of the HASP program generator.

2.29 At the program generation stage, the automatic diagnostic procedure in the HASP program generator is carried out to check the validity of component models and ensure that the component models have been selected correctly[8]. If the user has defined a circuit which is unacceptable, PGCOMP produces a diagnostic error message then returns to information interactive status to allow the user to reconsider. If the "model details" is acceptable, the program generator uses the information gained from COMPON.DAT, termed the component model attributes file, to create a correct calling sequence in the system model segment AUX.FOR and input control segment CONTRL.FOR by a special "SORT" algorithm in the subroutine PGCOMP.

Sort Process of Calling Sequence of Component Models

2.30 A correct calling sequence of component models in the simulation ensures that the output of each component model for time t is computed on the basis of the input values of this model for time t. An incorrect calling sequence would update a model's output for time t with input values for time $t-dt$. In this case, a time lag produced by this incorrect calling sequence could seriously affect stability and accuracy of the solution, even make the whole simulation failed. When a "model

details" of component models is determined by the user in any order, therefore, a correct calling sequence in simulation programs should be one in which the output of each component model in the sorted sequence can be computed using the known values. In order to ensure that the simulation of hydraulic systems can correctly be completed using HASP, a correct calling sequence for component models required must be arranged in a system model subroutine AUX and a parameter input control subroutine CONTRL and its form is respectively:

in AUX subroutine:

```
CALL TK01(...
CALL PU00(...
CALL PM00(...
:
:
```

or in CONTRL subroutine:

```
CALL TK01IN(...
CALL PU00IN(...
CALL PM00IN(...
:
:
```

2.31 **Sorting Process.** The HASP sorting algorithm first determines an appropriate calling sequence for all instantaneous memory component models that contribute to the input of other component models. Next, it determines the calling sequence for other instantaneous component models necessary to obtain the input from the dynamic models described by differential equations (usually this kind of instantaneous model has two or more than two external links). Finally, the calling sequence for remaining dynamic component models is sorted by the program PGCOMP by means of the input of other models.

2.32 **Explanation of sort process.** In order to describe the sort process of a correct calling sequence of component models, consider the simple open loop hydrostatic transmission employed earlier and list its "model details" again as

follows:

```
09
TK0101 01
PU0001 01 02 03
PM0101 03
PI0501 02 04 05
PC0101 04 06
TK0102 06
MO0001 05 07 08
TK0103 07
LR0001 08
```

At the beginning of sort process, the memory models (dynamic and instantaneous) and non-memory models are determined by the sorting program PGCOMP of the HASP program generator according to the attribute details for each component model in this circuit. The attribute details of component models are supplied by the model attribute file COMPON.DAT. For instance, the attribute information of the tank model TK01 in the model attributes file COMPON.DAT is:

```
TK01
110000100N0
```

The first five digits on the second line mean:

(i) first digit-1. The model has a minimum of 1 external link.
(ii) second digit-1. The model has a maximum of 1 external link.
(iii)third digit-0. The model has no internal links.
(iv) fourth digit-0. The model has no signals.
(v) fifth digit-0. The model has no state variables.

The attribute information of other component models in this circuit are supplied below by the COMPON.DAT file:

a) pump model PU00

PU00
330001301N0

b) prime mover model PM00

PM00
110000100N0

c) pipe model PI05

PI05
180011203N1

d) relief valve model PC01

PC01
220000300N0

e) hydraulic motor model MO00

MO00
330001301N0

f) rotary load model LR00

LR00
110010701N2

According to above model attribute details, the correct calling sequence for the subroutines AUX and CONTRL is obtained using the sort process described below. This is performed in three steps.

2.33 **First step: Determine the calling sequence for all instantaneous memory component models.** The concept of memory model has been described previously in paragraph 2.3. Here the instantaneous memory component model is that whose output values are known, such as tank, prime mover and duty cycle models etc. Normally instantaneous memory component model only has one external link. Hence the sorting program PGCOMP can find two memory instantaneous models in the system by checking the model attribute details given above. Similarly, three non-memory instantaneous models and two dynamic models in circuit are also found by the sorting program PGCOMP. Here the models TK01 and PM00 are instantaneously memory because they have no state variables and only have one external links respectively. PU00, MO00 and PC01 are non-memory models because they have no state variables and at least have two external links. The models PI05 and LR00 are dynamic because they have state variables on their external links.

2.34 At the start of the sort process, the numbers associated with the models are identical to those supplied by the user for the practical circuit and appear as in table 1 below. This is termed the "model sort table".

TK0101	1		
PU0001	1	2	3
PM0001	3		
PI0501	2	4	5
PC0101	4	6	
TK0102	6		
MO0001	5	7	8
TK0103	7		
LR0001	8		

table 1

In this circuit, the tank model TK01 and the prime mover model PM00 are instantaneous memory ones, hence the outputs of these models are known values. The relevant external link numbers for models TK01 and PM00 are replaced by digit-0. The model sort table now appear as in table 2 below.

TK0101	0		
PU0001	1	2	3
PM0001	0		
PI0501	2	4	5
PC0101	4	6	
TK0102	0		
MO0001	5	7	8
TK0103	0		
LR0001	8		

table 2

2.35 Since the outputs of dynamic models PI05 and LR00 are also known values on their external links, relevant links on other non-dynamic component models also put 0 digit, as shown in model sort table 3.

TK0101	0		
PU0001	1	0	3
PM0001	0		
PI0501	2	4	5
PC0101	0	6	
TK0102	0		
MO0001	0	7	0
TK0103	0		
LR0001	8		

table 3

For instance, the outputs of model PI05 on the links 2,4 and 5 transfer to models PU00, PC01 and MO00 respectively, thus the link 2 on the model PU00, and the link 4 on the model PC01 as well as the link 5 on the model MO00 are replaced by zeros.

Afterwards, the subroutine PGCOMP gives the calling sequence of instantaneous memory component models according to following processes (1) to (4):

					"calling list"
PU0001	0	0	3		
PM0001	0			(1)	TK0101
PI0501	2	4	5	====>	
PC0101	0	6			
TK0102	0				
MO0001	0	7	0		
TK0103	0				
LR0001	8				

table 4

2.36 Once all external links of a component model are entirely replaced by zeros, then all the inputs to the corresponding components are known. Hence it can be added to the calling list. In the example under consideration, the first memory component model added to the calling list is TK0101. When such a component is found, it is put in the calling list. Meanwhile, the relevant input links of other component models linked with this model are replaced by zero. Here the link of TK0101 is 1 and hence the link 1 of PU0001 is replaced by zero. This operation is shown in model sort table 4.

Similarly, instantaneous memory models are added to the calling list one by one according to following processes.

					"calling list"
PU0001	0	0	0	(2)	TK0101
PI0501	2	4	5	====>	PM0001
PC0101	0	6			
TK0102	0				
MO0001	0	7	0		
TK0103	0				
LR0001	8				

table 5

					"calling list"
PU0001	0	0	0	(3)	TK0101
PI0501	2	4	5	====>	PM0001
PC0101	0	0			TK0102
MO0001	0	7	0		
TK0103	0				
LR0001	8				

table 6

					"calling list"
PU0001	0	0	0	(4)	TK0101
PI0501	2	4	5	====>	PM0001
PC0101	0	0			TK0102
MO0001	0	0	0		TK0103
LR0001	8				

table 7

2.37 **Second step: Determine the calling sequence for other instantaneous non-memory component models necessary to obtain the inputs to the dynamic models described by differential equations.** The components remaining at the left part of the model sort table 7 is sorted again. Since the links of models PU00, PC01 and MO00 are already occupied by zeros, they can be added to the calling list one by one in terms of the rules specified above. The sort process of the second step is completed by the processes (5) to (7).

					"calling list"
PI0501	0	4	5	(5)	TK0101
PC0101	0	0		====>	PM0001
MO0001	0	0	0		TK0102
LR0001	8				TK0103
					PU0001

table 8

After the model PU00 is put in the calling list, the relevant input of the model PI05 on link 2 is replaced by zero. this operation is shown in model sort table 8. And then the model PC01 is added to the calling list and the input of model PI05 on link 4 is replace by zero, which is shown as follows in model sort table 9:

					"calling list"
PI0501	0	0	5	(6)	TK0101
MO0001	0	0	0	====>	PM0001
LR0001	8				TK0102
					TK0103
					PU0001
					PC0101

table 9

And then the component model MO00 is put in the calling list in a same way and the inputs of LR00 and PI05 on the links 8 and 5 are replaced by zeros. This operation is shown in model sort table 10.

					"calling list"
PI0501	0	0	0	(7)	TK0101
LR0001	0			====>	PM0001
					TK0102
					TK0103
					PU0001
					PC0101
					MO0001

table 10

2.38 **Third step: Determine the calling sequence of remaining component models.** Since all external links of two component models remained have been entirely replaced by zeros, the calling sequence of two models is arranged in arbitrary order below.

PI0501
LR0001

Therefore, a correct calling sequence of models for an open loop hydrostatic transmission is eventually obtained below.

<u>Correct model calling sequence for a hydrostatic transmission</u>

TK0101
PM0001
TK0102
TK0103
PU0001
PC0101
MO0001
PI0501
LR0001

2.39 For the above example under the analysis, consider what would happen if the hydraulic pipe model PI05, a dynamic memory model, is replaced by a hypothetical instantaneous pipe model PIXX which is a non-memory component model. The link diagram of model PIXX is shown in figure 2.17, and it would provide pressure P in return for flow rates Q_i. If this model is connected to an instantaneous hydraulic pump model PU00, an instantaneous motor model MO00 and an instantaneous relief valve model PC01, the link diagram would appear as in figure 2.18. Since both P and Q_i are not state variables, PIXX01 cannot be called before PU0001, MO0001 or PC0101 because it requires Q_i, the flow rates, as input and these are given by PU0001, MO0001 and PC0101. Similarly PU0001, MO0001 or PC0101 also cannot be called before PIXX01 because they require P as input. In this case, the relationship of input and output information between two models such as PU00 and PIXX, PIXX and MO00 or PIXX and PC01 is implicit and it is impossible to determine which should be called first during the arrangement of a correct calling sequence. For this reason, the diagnostic procedure of sorting algorithm routine PGCOMP will give a diagnostic message "IMPLICIT RELATIONSHIP BETWEEN MODELS" and then stop the sort process. However, this kind of difficulty caused by linking non-memory component models could be overcome by using the approaches to solution of linking difficulties proposed in the paragraph 2.16 and the relevant sort examples are given below respectively.

2.40 **Case 1: Using a Combination Model to Solve the above Linking Difficulties.** In the simple open loop hydrostatic transmission discussed above, the hydraulic pipe model, pump model, relief valve model and the hydraulic motor model can be combined into one. The algebraic equations for four of them can be solved simultaneously in the resulting combination model. For instance, this combination model is called as CMPI, and its link diagram is shown in figure 2.19 and a link diagram for this circuit appears as in figure 2.20. In this case, except the model CMPI, other models in this circuit are memory ones, and hence the implicit relationship between models disappears and a correct calling sequence for the component models of this system can be obtained as follows by means of the sorting process described in paragraph 2.31.

Correct model calling sequence for case 1

TK0101
PM0001
TK0102
TK0103
CMPI01
LR0001

2.41 **Case 2: Using the pseudo-state variable method to solve the above linking difficulties.** A pipe instantaneous model PIA5 with an iterative procedure is chosen to replace the dynamic model PI05. In the model PIA5, the fluid compressibility is ignored, and hence pipe pressure P is an algebraic variable and the model equation is a non-memory instantaneous model originally. In order to avoid the linking difficulty caused by the implicit relationship between non-memory models, pressure P is made a pseudo-state variable whose derivative is zero and its value is computed instantaneously by the iterative technique described in Chapter 4. Since the pressure P is set as a pseudo-state variable, the implicit relationship between PIA5 and its adjacent non-memory models can be broken. This method proposed in the paragraph 2.19 ensures that the pressure P is still a state variable on the external links of the model, therefore, the model PIA5 becomes a memory instantaneous model which can supply known values to adjacent models. In this case, the model PIA5 will be identified as a dynamic memory model by the HASP sorting algorithm. Obviously, a correct calling sequence of component models of the system for case 2 should be identical with previous situation in which the dynamic pipe model PI05 is adopted and it is listed below.

Correct model calling sequence for case 2

TK0101
PM0001
TK0102
TK0103
PU0001
PC0101
MO0001
PIA501
LR0001

2.42 In a word, the general linking difficulties of hydraulic component models can be overcome by using special approaches proposed in this chapter, such as the state variable method, pseudo-state variable method, combination model method, etc. Once the linking difficulties are removed, as long as the input information of each component model in the model attributes file COMPON.DAT is valid, a correct calling sequence for any hydraulic system to be simulated will be obtained without any trouble by the sorting algorithm of the HASP program generator.

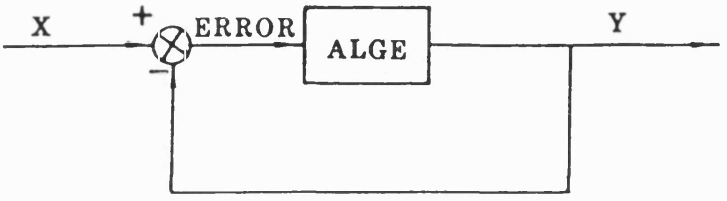


Figure 2.1 An implicit algebraic loop

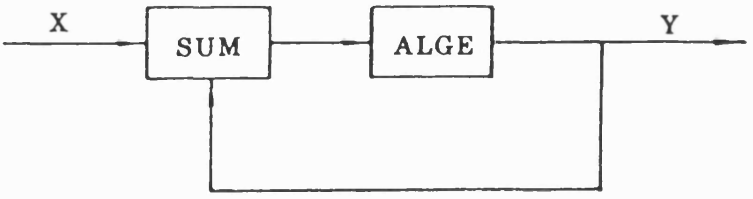


Figure 2.2 Non-memory algebraic loop

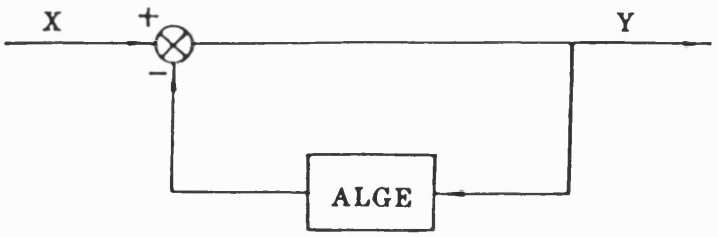


Figure 2.3 An implicit loop with an algebraic feedback

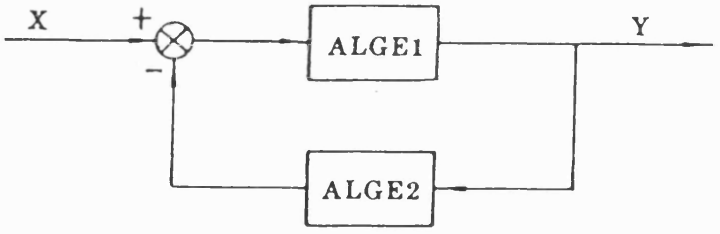


Figure 2.4 An implicit loop with algebraic forward and feedback

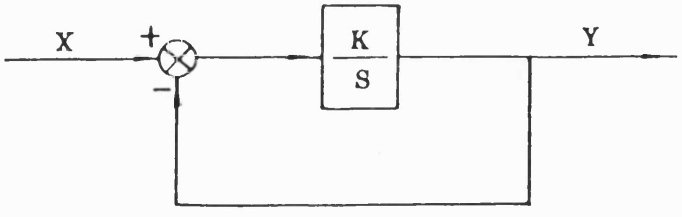


Figure 2.5 Integration loop

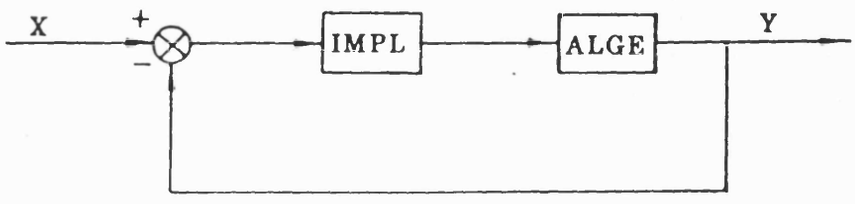


Figure **2.6** Iteration loop

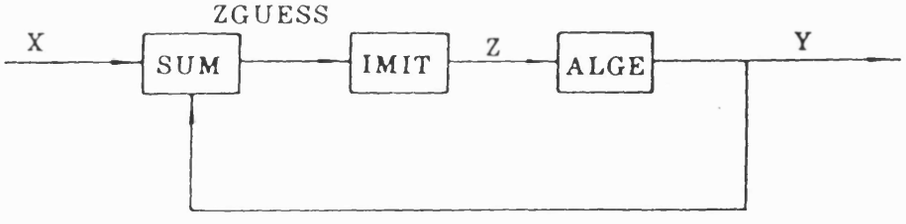


Figure 2.7 Block diagram for iteration loop

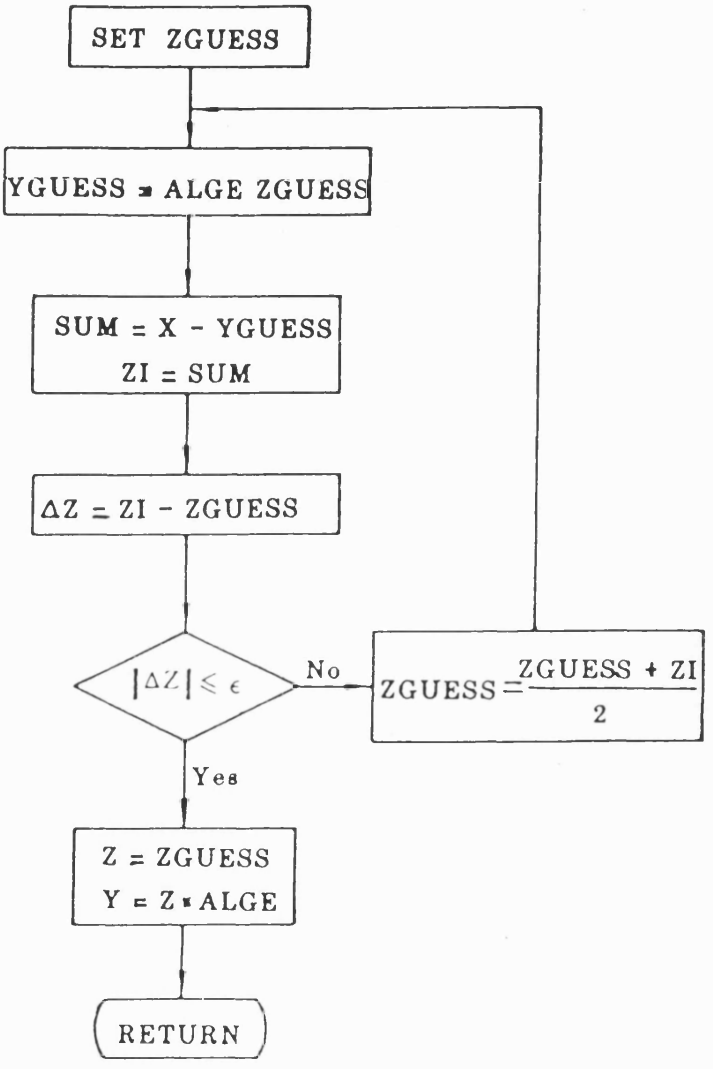


Figure 2.8 Iterative algorithm for breaking an implicit algebraic loop

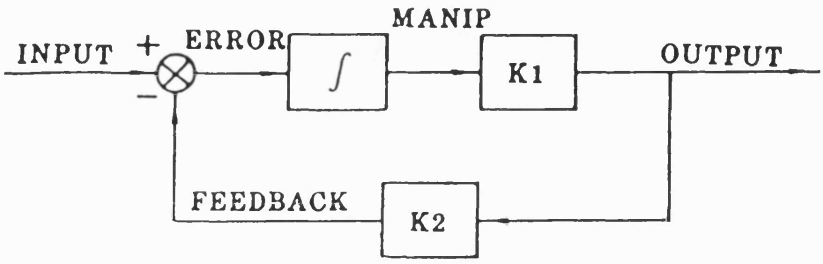


Figure 2.9 A simple feedback control system

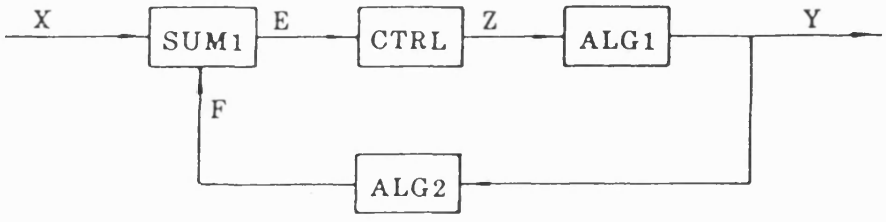


Figure 2.10 Block diagram of feedback control system

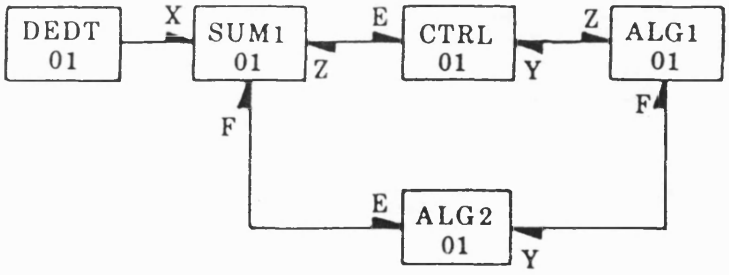


Figure 2.11 HASP power bond linking diagram

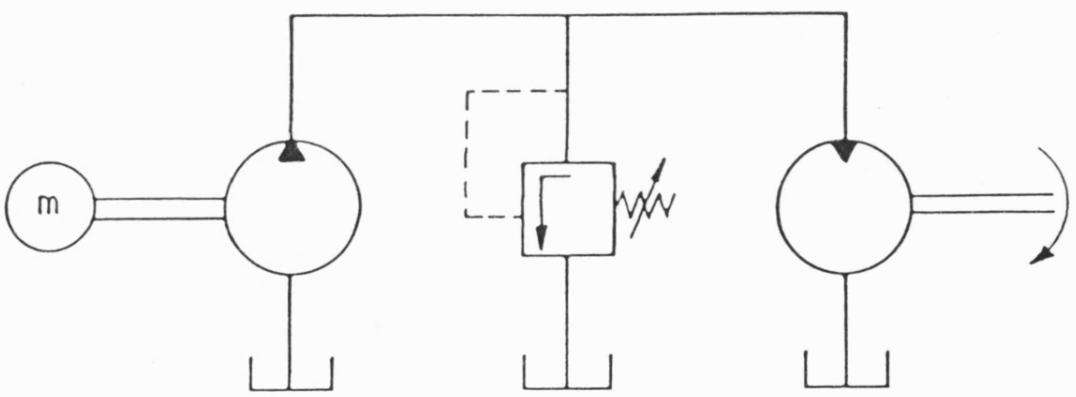


Figure 2.12 An example of an hydraulic circuit to be simulated

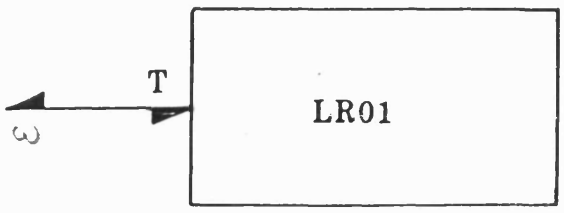


Figure 2.13 Link diagram for LR01

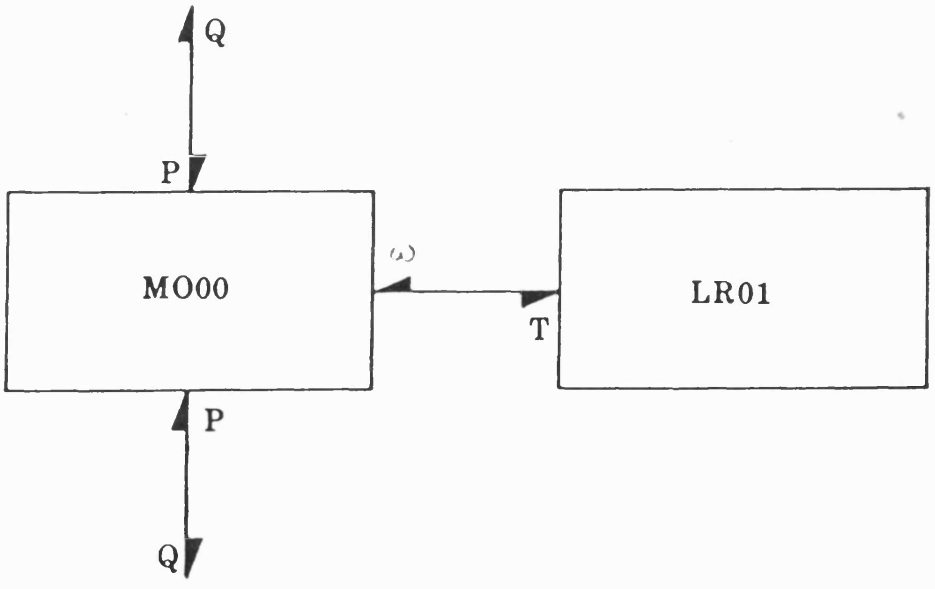
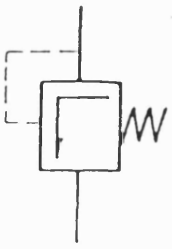
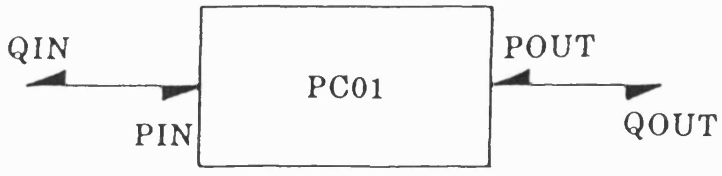


Figure 2.14 Power bond diagram showing LR01 connected to MO00



(a) CETOP symbol for a pressure relief valve



(b) Link diagram for instantaneous relief valve model PC01

Figure 2.15 Model of a relief valve

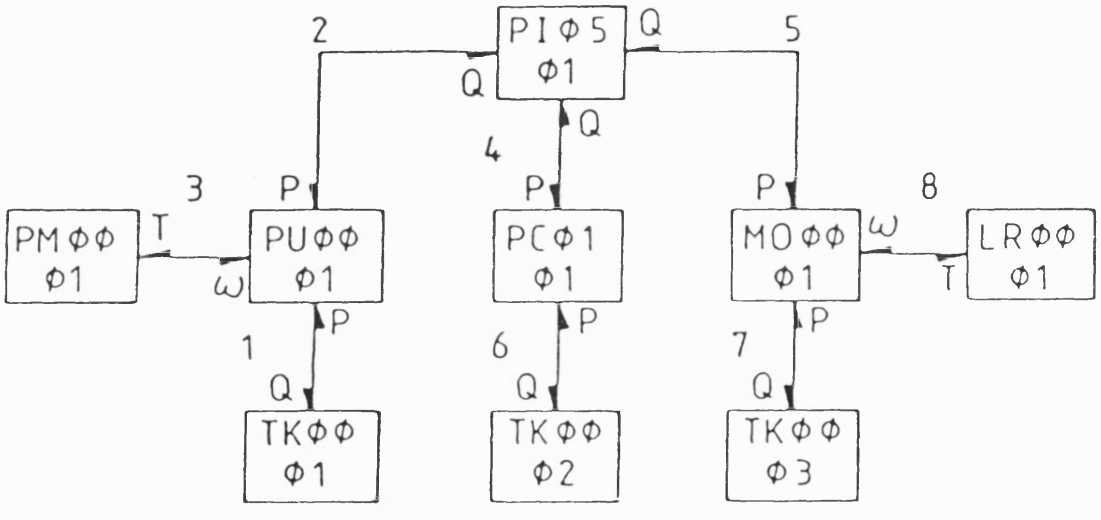


Figure 2.16 The circuit linking diagram corresponding to figure 2.12

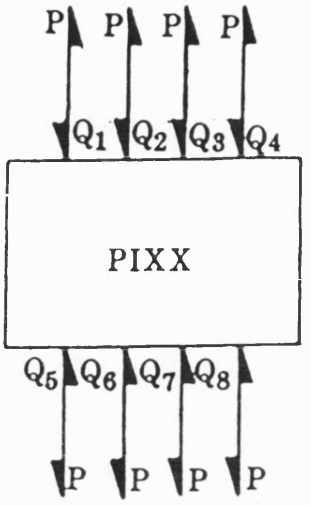


Figure 2.17 Link diagram for PIXX

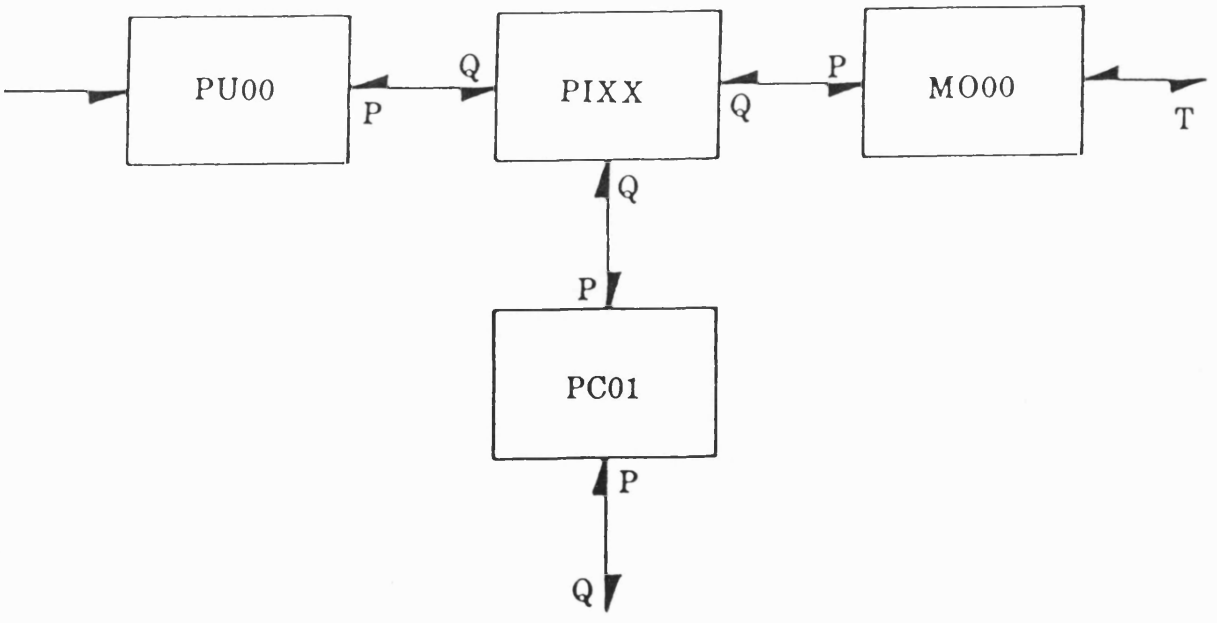


Figure 2.18 Power bond diagram showing PIXX connected to PU00, PC01 and MO00

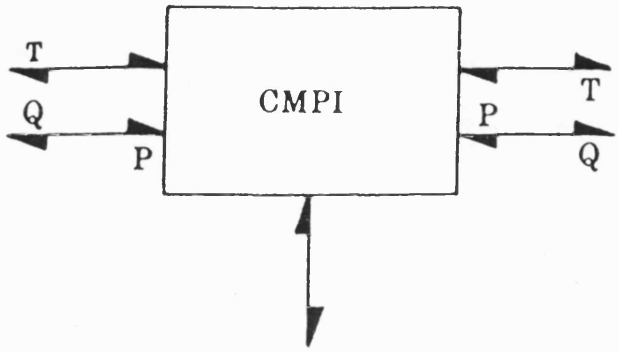


Figure 2.19 Link diagram for combination model CMPI

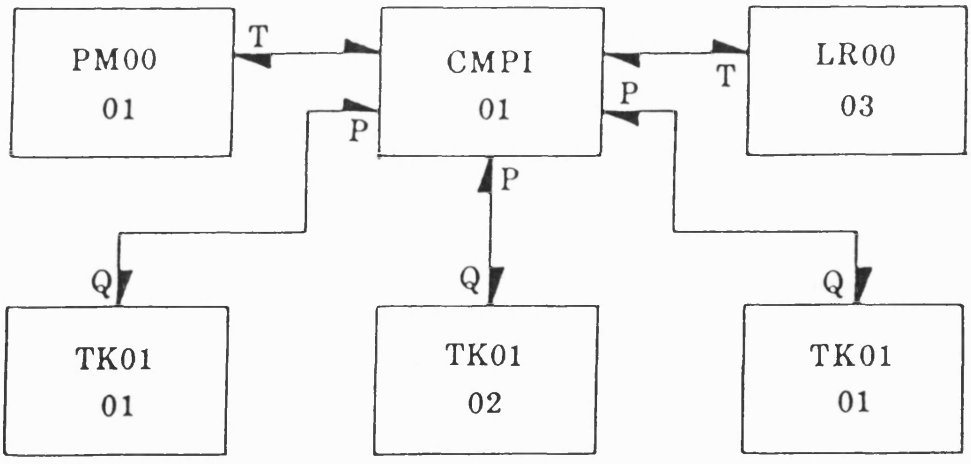


Figure 2.20 A circuit linking diagram with CMPI for a simple open loop hydrostatic transmission

CHAPTER 3

THE INVESTIGATION OF COMMON DIFFICULTIES IN SIMULATION OF HYDRAULIC SYSTEMS

INTRODUCTION

3.1 Mathematical stiffness makes a severe problem for the stability and accuracy of solutions for the integration method being used. In this case, the system simulation might easily fail, at least a very small simulation time step would be required, resulting is long computational time. Standard integration subroutines, specially multi-step integration methods, also might fail to simulate the dynamic performance of a system with discontinuities when the points of discontinuity are encountered in the simulation process. This Chapter gives an investigation and analysis of mathematical stiffness and discontinuity problems in order to solve these difficulties in the simulation of hydraulic systems.

INVESTIGATION OF MATHEMATICAL STIFFNESS DIFFICULTIES

3.2 **Mathematical Stiffness Problem.** Consider a very simple system of two first order differential equations[29].

$$\frac{dy_1}{dt} = -y_1$$

$$\frac{dy_2}{dt} = -100\, y_2 \qquad \ldots\ldots (3.1)$$

where initially $y_1 = y_2 = 1$ at $t = 0$. The analytical solution to this problem is readily seen to be:

$$y_1(t) = e^{-t}$$

$$y_2(t) = e^{-100t} \qquad \ldots\ldots (3.2)$$

The two coefficients of the exponents -1 and -100 are known as the eigenvalues λ_1 and λ_2.

3.3 A system of differential equations is said to be stiff if the real parts of the eigenvalues are negative and of widely differing magnitude. The mathematical stiffness of the system is generally measured by the ratio of the numerically largest real eigenvalue to the numerically smallest non-zero real eigenvalue so that for the above example, the stiffness ratio is 100. In this case, we hope to integrate this problem numerically with a reasonably large step size. However, according to the criterion of the stability, in order to obtain the stable solution of the system, the high order numerical methods and an extremely small value of h (time step) are required over the entire range of integration. As a result, the computational time required to integrate the system equation becomes excessive.

3.4 **Stiffness Definition** The initial value problem

$$\frac{dy}{dt} = f(t, y), \qquad y(0) = y_0 \qquad \ldots\ldots (3.3)$$

is stiff if following two conditions are satisfied

$$(i)\ Re(\lambda i) < 0 \qquad (i = 1,2,\ldots\ldots,s) \qquad \ldots\ldots (3.4)$$

$$(ii)\ S(t) = \frac{Max\left|Re(\lambda_i)\right|}{Min\left|Re(\lambda_i)\right|} >> 1 \quad (i = 1,2,\ldots,s) \qquad \ldots\ldots (3.5)$$

where the λ_i are the eigenvalues of $\frac{\partial f}{\partial y}$ evaluated on the solution $y(t)$ at t. $Re(\lambda_i)$ are the real parts of λ_i and $S(t)$ is the system stiffness ratio. In linear systems, the stiffness ratio is constant; however, in non-linear systems, such as a two stage relief valve and other examples considered in this thesis, it will vary with the operating point of the system.

For the above example, the system stiffness ratio $S(t)$ is 100.

Analysis of Stiffness Difficulty in Simulation of Hydraulic Systems

3.5 The analysis of mathematical stiffness difficulty in the simulation of hydraulic systems will be undertaken with the aid of a practical example. Consider a simple pump actuator circuit shown in figure 3.1. The fixed displacement pump is assumed to be lossless and runs at constant speed. The constant flow produced by the pump is directly supplied to the actuator which is moving at constant speed initially via a pipe in which the fluid is compressible. If a step increase of external load force F_a is applied to the actuator, it will decelerate and the system pressure rise. A relief valve is used to limit the system pressure if the external load becomes excessive. In order to simplify the analysis, the pump and actuator are assumed to exhibit no slip flow loss and constant friction force. The mathematical models to describe the physical behaviour of the system are given below.

(a) pump flow rate

$$Q_p = D_p \, \omega_p \qquad \ldots\ldots (3.6)$$

where D_p - pump displacement
ω_p - pump shaft angular speed

(b) flow rate through relief valve

$$Q_r = \frac{P - P_c}{k} \qquad \ldots\ldots (3.7)$$

where P_c - cracking pressure of relief valve
k - flow coefficient of relief valve
P - fluid pressure in the pipe

(c) actuator flow

$$Q_a = A \, v \qquad \ldots\ldots (3.8)$$

where A - area of actuator piston
v - speed of actuator piston

(d) rate of change of pressure in the pipe

$$\frac{dP}{dt} = \frac{\beta}{V} (Q_p - Q_r - Q_a) \qquad \ldots\ldots (3.9)$$

where β - fluid bulk modulus

V - combined pipe and actuator volume on the piston side

(e) motion equation

$$\frac{dv}{dt} = \frac{1}{M} (P A - f_v v - F_a - F_f) \qquad \ldots\ldots (3.10)$$

where M - actuator mass

f_v - viscous friction coefficient

F_a - external load force of actuator

F_f - friction force

Therefore

$$\frac{dP}{dt} = \frac{\beta}{V} (D_p \omega_p - \frac{(P - P_c)}{k} - A v) \qquad \ldots\ldots (3.11)$$

$$\frac{dv}{dt} = \frac{1}{M} (P A - f_v v - F_a - F_f) \qquad \ldots\ldots (3.12)$$

or more concisely by the 2 by 2 matrix differential equation

$$\begin{bmatrix} \frac{dP_1}{dt} \\ \frac{dv}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{\beta k}{V} & -\frac{\beta A}{V} \\ \frac{A}{M} & -\frac{f_v}{M} \end{bmatrix} \times \begin{bmatrix} P \\ v \end{bmatrix} + \begin{bmatrix} \frac{\beta D_p \omega_p}{V} + \frac{\beta P_c}{Vk} \\ -\frac{F_a + F_f}{M} \end{bmatrix}$$

This 2 x 2 matrix differential equation is of the general form

$$\dot{Y} = A\ Y + B \qquad \ldots\ldots (3.13)$$

where $A\ Y$ is a vector, known as the complementary function, which represents the transient behaviour of the system. And B is a vector, known as the particular integral, which defines the steady state behaviour of the system.

3.6 Associated with every square matrix A there is a special set of vectors, called eigenvectors, and a related set of scalars, called eigenvalues.

The eigenvalues, also known as characteristic values, are the solutions λ of,

$$(A - \lambda I)Y = 0$$

where I is a unit diagonal matrix, The values of λ are evaluated from the determinant

$$|A - \lambda I| = 0$$

From above matrix the determinant is given by:

$$\begin{vmatrix} -\frac{\beta}{Vk} - \lambda & -\frac{\beta A}{V} \\ \frac{A}{M} & -\frac{f_v}{M} - \lambda \end{vmatrix} = 0 \qquad \ldots\ldots (3.14)$$

which on expansion gives:

$$\lambda^2 + \lambda \left(\frac{f_v}{M} + \frac{\beta}{Vk}\right) + \frac{\beta}{MV}\left(\frac{f_v}{k} + A^2\right) = 0$$

where the flow coefficient of relief valve k is small, and the pipe volume and the piston area are very small. Although usually the actuator mass M and the viscous load are large, the term $\frac{f_v}{M}$ may be considered negligible compared with the term $\frac{\beta}{Vk}$ because the fluid bulk modulus β is very large, and the term A^2 also is negligible compared with the term $\frac{f_v}{k}$, and hence the eigenvalue equation of the system can be rearranged as follows:

$$\lambda^2 + \lambda \frac{\beta}{Vk} + \frac{\beta f_v}{MVk} = 0$$

and the solutions of system eigenvalues are given by:

$$\lambda = -\frac{\beta}{2Vk} \pm \frac{1}{2}\left|\frac{\beta^2}{V^2k^2} - \frac{4\beta f_v}{MVk}\right|^{0.5}$$

and the stiffness ratio of the system can be expressed below.

$$S = \frac{1 + \sqrt{1 - \frac{4Vkf_v}{M\beta}}}{1 - \sqrt{1 - \frac{4Vkf_v}{M\beta}}} \qquad \ldots\ldots (3.15)$$

Where k can be very small indeed and in special cases can be zero.

3.7 From above expression of the system stiffness ratio, the most important parameters which affect stiffness ratio might be the hydraulic pipe volume V and the relief valve flow coefficient k because other parameters such as the actuator mass M, viscous coefficient f_v and the fluid bulk modulus β have no big change in practical hydraulic systems. Suppose the mass M, viscous coefficient f_v and bulk modulus β are constant values, the smaller the pipe volume V or relief valve flow coefficient k, the larger the denominator of above expression of stiffness ratio S, and the smaller the numerator of S. It is possible that the stiffness ratio S of the system would be very large if the hydraulic pipe volume V were extremely small or the gradient of flow pressure characteristics $\frac{1}{k}$ were extremely high, in other word, the mathematical stiffness of this system would be extremely serious. Therefore, in the simulation of this hydraulic system, the main mathematical stiffness is produced by the hydraulic pipe volume and the gradient of relief valve steady state flow pressure characteristics. For instance, taking following practical parameters respectively:

<u>Test parameters of a pump-actuator</u>

$\beta = 1.8x\,10^4$ bar

$M = 870\,kg$

$k = 0.1\,min/l$ bar

$f_v = 1000\,N/(m/s)$

$A = 25\times10^{-4}\,m^2$

Case 1: V = 100 litre = $10^{-1}\ m^3$

obtains the solution of the system stiffness ratio as follows:

$$S(t) = \frac{|\ \mathrm{Re}(\lambda_2)\ |}{|\ \mathrm{Re}(\lambda_2)\ |} = 9.37724\times10^{9}$$

Case 2 : V = 0.1 litre = $10^{-4}\ m^3$

The solution of S is given by:

$$S(t) = \frac{|\ \mathrm{Re}(\lambda_2)\ |}{|\ \mathrm{Re}(\lambda_2)\ |} = 9.377615\times10^{12}$$

Obviously, the mathematical stiffness ratio S of this linear actuator system is very large as the pipe volume is very small. According to the definition of the mathematical stiffness given earlier, the system model described this linear actuator circuit is termed to be mathematically stiff. The need for very small step lengths to handle stiff system case will obviously mean much longer simulation run times.

Techniques for Coping with Mathematical Stiffness of Hydraulic Systems

3.8 Numerical Integration Methods for Stiff Systems. In order to solve the mathematical stiff problems in the simulation of hydraulic systems, several numerical integration methods such as Gear's method, have been developed[22],[33] and the Gear method has been implemented in the HASP simulation package. A multi-step integration formula is one which uses information from more than one previous integration step in order to obtain a numerical solution to a differential equation. Gear's method is a widely used multi-step method currently available for solving stiff problems and is employed for HASP, and a description about this method is given by Tomlinson[15]. Multi-step methods are known to have excellent stability properties and are computationally efficient because they use information

prior to as well as at the current integration step. Single step algorithms such as variable time step Runge-Kutta-Merson and Runge-Kutta-Fehlberg methods where only information from the current integration step is used to obtain the solutions of differential equations are not usually considered suitable for solving stiff problems, but it will be shown in this thesis that they are adequate for many of the systems met with. A detail description of single step algorithms is given in Chapter 6.

3.9 Although these numerical integration methods can solve most of the stiff problems encountered in the simulation of fluid power systems, they still could not solve some very serious stiff problems. For instance, suppose the volume of a pipe in above linear actuator circuit described is extremely small, and then the mathematical stiffness ratio is extremely high. Therefore, the operation of numerical integration methods could fail or could take a long long time to run a simulation. Another example is that of an orifice between two pipelines, let zero flow dP/dQ=0 and under these conditions any system will exhibit infinite stiffness. One method of avoiding this, used in orifice models in HASP[8], is to assume a small linear region about the $Q = 0$ axis. Another is to use the pseudo-analytical technique proposed by Bowns and Rolfe[30].

3.10 **An iterative technique for avoiding mathematical stiffness of hydraulic systems.** Consider a system involving a restriction linking two pipes whose schematic is shown in figure 3.2. Since the fluid volume of a pipe between two restrictor valves is very small comparing with that in other two pipes connected with restrictors, the mathematical stiffness problem of this system would be very serious. The method of avoiding this, proposed by the author, is to assume that the fluid compressibility of a pipe between two restrictor valves is negligible, and then the pressure in the pipe is computed instantaneously by an iterative technique termed half-interval iterative method. A full description of this iterative technique is shown in chapter 4.

INVESTIGATION OF DISCONTINUITY DIFFICULTIES

3.11 Although most of hydraulic systems are essentially nonlinear, this does not pose any serious difficulty in the system simulation in HASP[8][15]. However, the discontinuous nature of systems does cause difficulties in the numerical integration. Many standard integration routines such as standard Gear method, are prone to fail when they reach a point of discontinuity[15][32][18][35]. Normally, there are two types of discontinuity problems, one is the mathematically discontinuous problem and another is the physical discontinuous problem. The natures of both discontinuous problems are discussed below.

Mathematical and Physical Discontinuity Problems

3.12 Consider a simple system in the form of a first order set:

$$\frac{dy_i}{dt} = f_i(t, y_1, y_2, \ldots, y_n) \quad i = 1,2,\ldots,n \qquad \ldots\ldots (3.16)$$

In a system with discontinuities, the function, f_i will change according to the state of the system. Therefore to generalise (3.16) to permit m different states $S_1, S_2, \ldots, S_m$

$$\frac{dy_i}{dt} = f_{ij}(t, y_1, y_2, \ldots, y_n) \quad i = 1,2,\ldots,n$$
$$j = 1,2,\ldots,m \qquad \ldots\ldots (3.17)$$

where the state, S, of the system is determined by a set of discontinuity functions $\Phi_k(t, y_1, y_2, \ldots, y_n)$ which are defined such that a discontinuity occurs when one of the conditions $\Phi_k = 0$ is satisfied.

3.13 **Mathematical Discontinuities.** The system described by the equation (3.17) could be considered to be mathematically discontinuous if one of following two conditions is satisfied:

(a) a dependent variable jumps instantaneously from one value to another due to the change of the system state S.

(b) the time derivative of a dependent variable jumps instantaneously due to the change of the system state S.

3.14 **Physical Discontinuities.** The system described by the equation (3.17) is considered to be physically discontinuous if one of following conditions is satisfied:

(a) a dependent variable changes gradually due to a jump of its time derivative from one value to another.

(b) a dependent variable jumps instantaneously from one value to another due to the change of the system state S.

(c) the time derivative of a dependent variable jumps instantaneously due to the change of the system state S.

An example of a discontinuity could be an actuator or valve with end-stop problems. When the actuator or valve hits a stop, a discontinuous change of the state variables, including the displacement X, the velocity v and the acceleration a will occur and are demonstrated in figure 3.3.

3.15 Suppose the piston of a linear actuator moves in a constant speed, and then it stops at the end-stop position as it hits a stop. The displacement X and velocity v are shown in figure 3.3(a). According to above classification of a discontinuity, the v variable is mathematically discontinuous and the variables X and v are physically discontinuous. Figure 3.3(b) shows that the motion of the linear actuator is variable acceleration one, and hence the acceleration a is mathematically and physically discontinuous and the velocity v is physically discontinuous only. The

displacement X is continuous. Similarly, in figure 3.3(c), the displacement X and velocity v are continuous variables, but the acceleration a is a physically discontinuous variable. Other examples of physical discontinuities in hydraulic systems are discontinuous input force or flow as shown in figure 3.4.

Techniques for Coping with Discontinuous Problems

3.16 Both types of discontinuous problems can be overcome by employing special simulation techniques at the points of discontinuity.

3.17 **Cubic Smoothing Technique.** The discontinuous problems of mathematical models usually can be overcome by employing a cubic smoothing polynomial for a transition region used to represent the function between the points of discontinuity. For instance, in order to integrate across the discontinuities the Gear's integration routine was modified by Caney[34]. The present method used in HASP demands that the discontinuous functions in the function routine (AUX) are integrated piecewise in smooth sections by the inclusion of interval halving and restarting procedures. The discontinuous functions, or functions with discontinuous derivatives are divided into a series of continuous sections. At any integrated time the integration utilises the equations for one section only and it is essential that these are continuous. This method ensures that the integration algorithm always operates with continuous functions. For instance, consider the modelling of an instantaneous model of a pressure relief valve shown in figure 2.15. The flow from the valve outlet is approximated by the equations:

$$Q_{OUT} = 0, \qquad \text{if}(P_{IN} - P_{OUT} - P_C) \leqslant 0$$

$$Q_{OUT} = K\,(P_{IN} - P_{OUT} - P_C), \quad \text{if}(P_{IN} - P_{OUT} - P_C) > 0 \qquad \ldots\ldots (3.18)$$

where P_{IN} - valve inlet pressure in bar
P_{OUT} - valve outlet pressure in bar

P_c - cracking pressure in bar

ΔP - valve pressure drop in bar

Q_{IN} - valve inlet flow in l/s

Q_{OUT} - valve outlet flow in l/s

k - average steady state flow - pressure drop gradient in (l/s)/bar

A straightforward computerised translation of equation (3.18) would cause integration problems for Gear's integrator due to the discontinuity in $\frac{dQ}{d\,\Delta P}$ at the cracking pressure. For this reason, a cubic model is required to give a continuous flow and flow derivative function.

The flow in the region $P_c \leqslant \Delta P \leqslant (P_c + 0.01)$ bar is determined using a cubic polynomial.

$$x_1 = P_c$$

$$x_2 = P_c + 0.01 \text{ bar}$$

$$h = x_2 - x_1 \text{ bar} = 0.01 \text{ bar}$$

$$z = \frac{(x - x_1)}{h} = \frac{(P_{IN} - P_{OUT} - P_c)}{0.01}$$

$$F_1 = f(x_1) = 0$$

$$G_1 = f_1(x_1) = 0$$

$$F_2 = f(x_2) = k(\Delta P - P_c) = 0.01\,k$$

$$G_2 = f_2(x_2) = k$$

$$S = F_2 - F_1 = 0.01\,k$$

$$Q_1 = hG_1 = 0$$

$$Q_2 = hG_2 = 0.01\,k$$

$$A = Q_1 + Q_2 - 2S = -0.01\,k$$

$$B = 3S - 2Q_1 - Q_Z = 0.02\,k$$

The cubic polynomial is given by:

$$Q = A_z^3 + B_z^2 + Q_1 z + F_1$$

$$= -0.01k\left(\frac{P-P_c}{0.01}\right)^3 + 0.02\left(\frac{\Delta P - P_c}{0.01}\right)^2 \qquad \ldots\ldots (3.19)$$

The simulated model characteristics are shown in figure 3.5.

3.18 A detailed description of cubic smoothing technique is given by Tomlinson[15]. In the mathematical modelling of components, however, to ensure the continuity of equations, sometimes the use of cubic smoothing functions becomes excessive. Even some false force terms which is not very reasonable for end-stop problems are also added into mathematical model, for instance, the end-stop spring stiffness force and the end-stop damping force. In this case there are some present model calculation subroutines in HASP seem to be too complex, and sometimes simulation results obtained with few models are not satisfactory.

3.19 **Conditional Statement Technique.** The cubic smoothing technique is necessary for the discontinuous problems with multi-step methods. The discontinuous problems with one-step methods can be overcome by employing a conditional statement technique at the points of discontinuity. For instance, two established Runge-Kutta type integrators, one a fixed step RK4 and the other a variable step Kutta Merson technique, KUTMER, have been programmed in a form suitable for HASP. These are one-step integration methods for the solution of numerical initial value problems. Since the computed values of state variables at any time point t_{n+1} only depend on those at previous time point t_n rather than any intermediate calculation process between two time points, the effect of the discontinuities in model equations on integration results can be erased when conditional statements are introduced into the component model calculation

subroutines. It is very important, however, that all restrained conditions of the discontinuities in the equations have to be put into the conditional statement section of model subroutines and they must be defined and written by the writer of model subroutine. For instance, consider the modelling of an instantaneous model of a relief valve using conditional statement technique. The discontinuity region can be described by "IF STATEMENTS" as follows:

```
      IF (LIMIT.EQ.1) GOTO 111
      IF ((PIN-POUT-PC).LT.0.D0) THEN
        QOUT=0.D0
      ELSE
        QOUT=K*(PIN-POUT-PC)
      END IF
  111 CONTINUE
```

The operation of the conditional statement section is controlled by the integrator indicator "LIMIT". At the beginning of a simulation, LIMIT is set to 0. During the intermediate calculation process in the simulation, LIMIT is set to 1 and at the completion of a step it is set to 2. A detailed discussion about LIMIT is given in the paragraphs 6.27 and 6.41. A detailed description of HASP new integrators RK4, KUTMER and KUTFEH is shown in the Chapter 6 of this thesis.

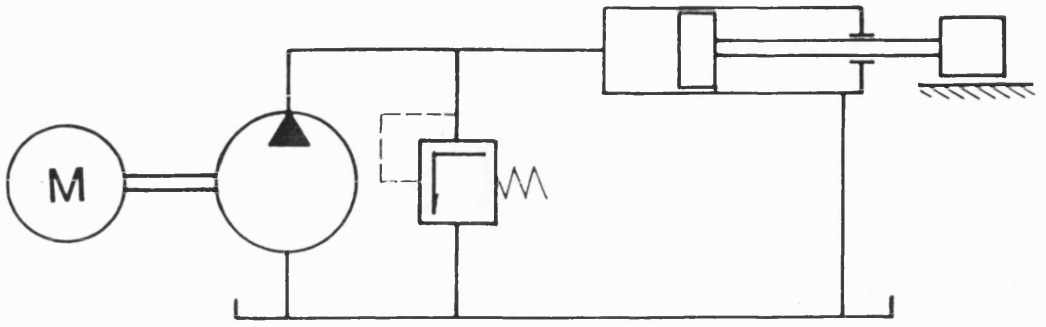


Figure 3.1 Hydraulic linear actuator system

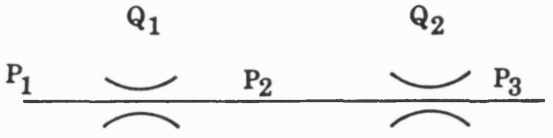


Figure 3.2 Simple pipe-orifice system

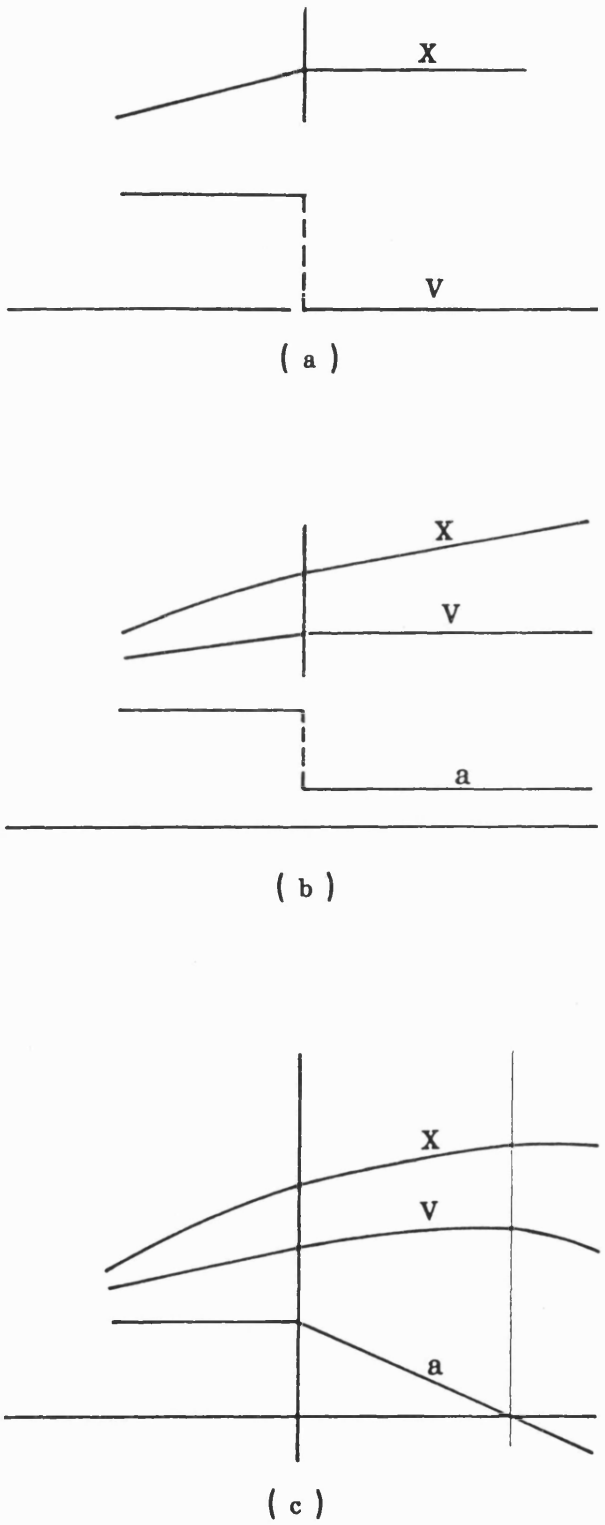
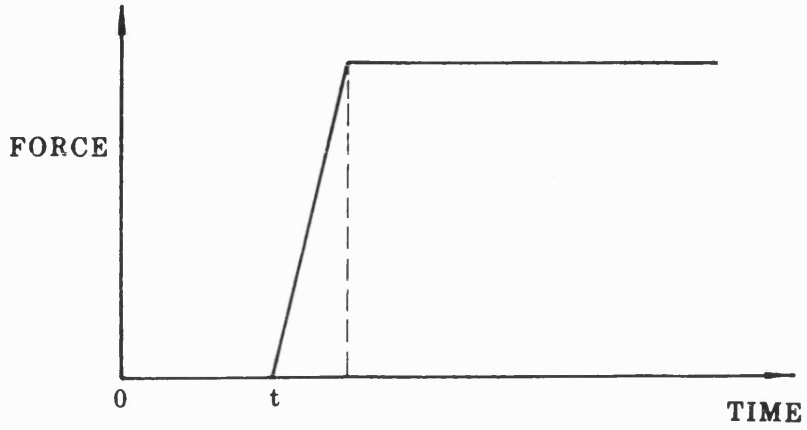


Figure 3.3 Discontinuity Examples



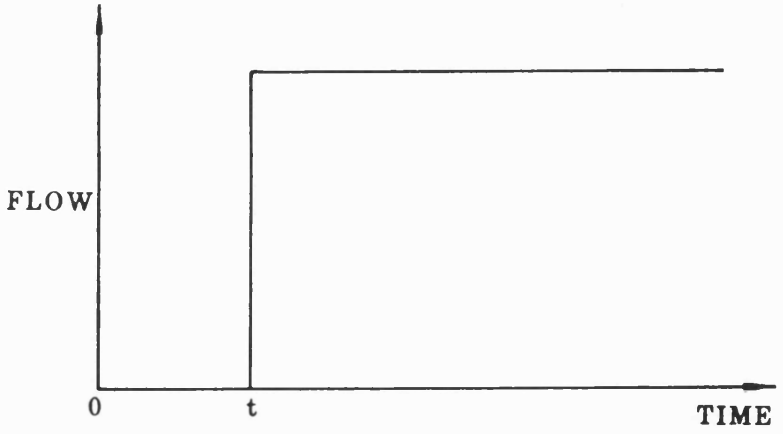


Figure 3.4 Discontinuous input force and flow rates

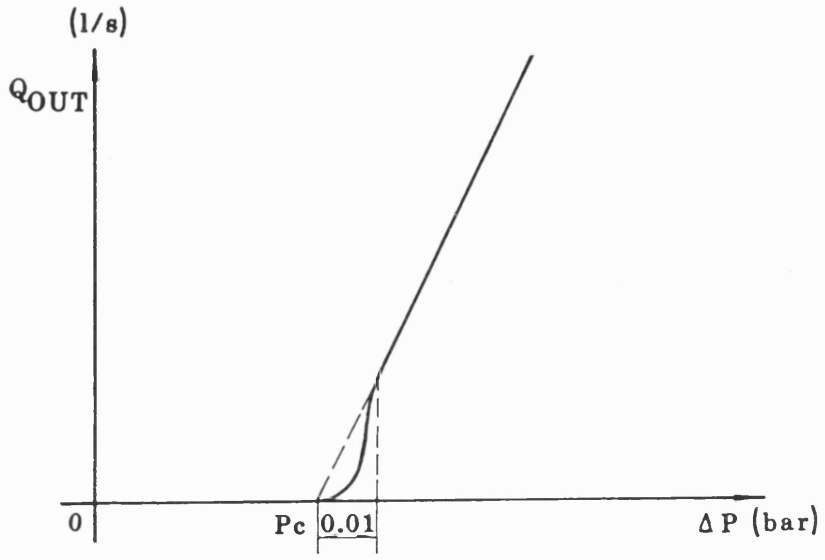


Figure 3.5 Simulated Model Characteristics

CHAPTER 4

ITERATIVE TECHNIQUE FOR AVOIDING STIFFNESS PROBLEMS IN THE SIMULATION OF HYDRAULIC SYSTEMS

INTRODUCTION

4.1 Mathematical stiffness often occurs in the simulation of hydraulic systems due to the wide variation of time constants which may be present. Sometimes some of the parameters can change at a very fast rate, leading to extreme computational difficulties. In some cases the change in a particular variable is so rapid that it would be sufficiently accurate to consider it as changing instantaneously.

4.2 One possible technique is to remove the state variable from the component model and replace the state equation by an algebraic equation. For example, in HASP we often ignore the dynamics of a relief valve because it can be considered instantaneous compared with the speed of system response.

In this case, a relief valve can be described by the pressure-flow instantaneous characteristic shown in figure 4.1, and relevant algebraic equations are given as follows:

$$Q_{OUT} = 0, \qquad \text{if}(P_{IN} - P_{OUT} - P_C) \leqslant 0 \qquad \ldots\ldots (4.1)$$

$$Q_{OUT} = k\,(P_{IN} - P_{OUT} - P_C), \quad \text{if}(P_{IN} - P_{OUT} - P_C) > 0$$

where P_{IN} - valve inlet pressure in bar
P_{OUT} - valve outlet pressure in bar
P_c - cracking pressure in bar
ΔP - valve pressure drop in bar

Q_{IN} - valve inlet flow in l/s

Q_{OUT} - valve outlet flow in l/s

k - average flow - pressure drop gradient during the 'open' period in (l/s)/bar

4.3 However, in the performance analysis or design of a two stage relief valve shown in figure 4.2, the dynamics of the valve may be very important. In this case, this relief valve including two poppet valves, one control orifice and one pilot chamber may be considered as a "system". Since the compliance of fluid in pilot chamber is very low comparing with that in inlet pipe of valve, the lag in pressure rise P_2 can be ignored. An instantaneous value of pressure for P_2 can be obtained by an iterative procedure rather than by solving the differential equation describing the pilot chamber pressure. As a result, the mathematical stiffness problem in the simulation of a two stage relief valve can be avoided.

This chapter describes iterative techniques used to model a two stage relief valve and also to model hydraulic pipes. It also contains a description of the relevant modelling techniques and tests carried out to ensure that the simulation is operational.

ITERATION USED TO MODEL A TWO STAGE RELIEF VALVE

4.4 **Mathematical Model of a Two Stage Relief Valve** Consider a two stage relief valve shown in figure 4.2, the inlet pressure P_{IN} is fed through a control orifice in the main poppet to act on the pilot. When the inlet pressure is high enough it lifts the pilot poppet off its seat, allowing flow to tank. The pressure drop across the control orifice lifts the main poppet, relieving the inlet pressure to tank.

4.5 The mathematical model describing a two stage relief valve generally involves the solution of high order differential equations, due to the complex nature of this hydraulic component. For the two stage relief valve shown in figure 4.2, a mathematical model is usually developed making the following assumptions[35]:

(i) the poppet valves have no chamfer on their seats.

(ii) the compressibility of the oil in the downstream chamber is neglected.

(iii) no radial forces exist on either of the poppet valves and lateral force acting on the valve is ignored.

(iv) transient flow forces acting on poppet valves are ignored.

(v) gravity effects are neglected.

(vi) flow is irrotational.

4.6 **Model Equations.** The mathematical model developed for this relief valve includes one flow continuity equation of the pilot stage, one motion equation of the pilot poppet and one motion equation of the main poppet. They are given below respectively.

a) Flow continuity equation of pilot stage

$$\frac{V_2}{\beta}\frac{dP_2}{dt} = Q_L + A_p v_x - Q_2 - A_2 v_y \qquad \ldots\ldots (4.2)$$

From this equation, the pressure behaviour of the pilot chamber can be described by:

$$\frac{dP_2}{dt} = \frac{\beta}{V_2}\left(Q_L + A_p v_x - Q_2 - A_2 v_y \right) \qquad \ldots\ldots (4.3)$$

where V_2 - fluid volume in pilot chamber

β - fluid bulk modulus

P_2 - pressure in pilot chamber

A_p - area of spring chamber of main stage

A_2 - area of pilot poppet seat

v_x - velocity of main poppet valve

v_y - velocity of pilot poppet valve

We will now assume that the flow through the orifice is proportional to the square root of the pressure drop across it, i.e.

$$Q_L = K_{or}\sqrt{(P_{IN} - P_2)}$$

where K_{or} - flow coefficient of orifice

P_{IN} - inlet pressure of two stage relief valve

Q_2 - flow through the orifice of pilot poppet

$$Q_2 = \pi C_d D_2 Y Sin(\alpha_2)\sqrt{2(P_2 - P_3)/\rho} \qquad \ldots\ldots (4.4)$$

C_d - coefficient of discharge

D_2 - diameter of pilot stage poppet seat

α_2 - angle of pilot poppet valve face

P_3 - pressure of contraction cross section

Y - displacement of pilot poppet

ρ - fluid density

b) Equation of motion for pilot poppet valve. Since the transient flow forces acting on the poppet valves are assumed to be ignored, the equation of motion for pilot poppet valve can be described by:

$$M_2 \frac{d^2Y}{dt^2} + f_2 \frac{dY}{dt} + K_2 Y = P_2 A_2 - P_{cr} A_2 - \frac{\rho Q_2^2}{C_d A_c} Cos(\alpha_2) \qquad \ldots\ldots (4.5)$$

where $\frac{\rho Q_2^2}{C_d A_c} Cos(\alpha_2)$ is steady state flow force acting on the pilot poppet valve. Assume that the areas of the contracted jet on opening and closing are equal in the

pilot poppet orifice, i.e.

$$A_{open} = \pi C_d D_2 Y Sin(\alpha_2) = A_c$$

and combining equations (4.4) and (4.5), we obtain

$$M_2 \frac{d^2Y}{dt^2} + f_2 \frac{dY}{dt} + K_{e2} Y = P_2 A_2 - P_{cr} A_2 \qquad \ldots\ldots (4.6)$$

where M_2 - mass of pilot poppet
f_2 - viscous coefficient of pilot stage
P_{cr} - cracking pressure of pilot stage
K_2 - spring stiffness of pilot stage
K_{e2} - effective spring stiffness

$$K_{e2} = K_2 + \pi C_d D_2 Sin(2\alpha_2)(P_2 - P_3)$$

c) Equation of motion for main poppet valve. Similarly, assume the transient flow force acting on the main poppet valve is ignored and the areas of the contracted jet on opening and closing are equal in the main poppet orifice, the equation of motion for main poppet valve can be expressed in the same form:

$$M_1 \frac{d^2X}{dt^2} + f_1 \frac{dX}{dt} + K_{e1} X = P_{IN} A_1 - P_2 A_p - F_1 \qquad \ldots\ldots (4.7)$$

where M_1 - mass of main poppet valve
f_1 - viscous coefficient of main stage
F_1 - spring preload of main poppet valve
A_1 - area of main stage poppet seat
K_{e1} - equivalent spring stiffness

$$K_{e1} = K_1 + \pi C_d D_1 Sin(2\alpha_1)(P_{IN} - P_3)$$

K_1 - spring stiffness of main stage

D_1 - diameter of main stage poppet seat

α_1 - angle of main poppet face

d) Flow continuity equation of inlet pipe. If we consider the dynamic effect of an inlet pipe, which is connected to the relief valve, the model of a two stage relief valve may also have a flow continuity equation of the inlet pipe which is given below.

$$\frac{V_1}{\beta}\frac{dP_{IN}}{dt} = Q_{IN} - Q_1 - Q_L - A_1 v_x \qquad \ldots\ldots (4.8)$$

Then the pressure behaviour of inlet pipe is described by:

$$\frac{dP_{IN}}{dt} = \frac{\beta}{V_1}\left(Q_{IN} - Q_1 - Q_L - A_1 v_x \right) \qquad \ldots\ldots (4.9)$$

where Q_{IN} - input flow

V_1 - the volume of the inlet pipe

Q_1 - flow through the orifice of main poppet

$$Q_1 = \pi C_d D_1 X \, Sin(\alpha_1) \sqrt{2(P_{IN} - P_3)/\rho}$$

X - displacement of main poppet valve

4.7 The mathematical model so obtained would result in a sixth order differential equation. The volume between the control orifice and the pilot stage is very small comparing with the external volume of the pipe in most cases. Thus the transient pressure changes of equation (4.3) are much more rapid than those of the remainder of the system and long simulation times are inevitable no matter what technique is used to integrate the equations.

Iterative Technique

4.8 Simulation times can be speeded up if the compliance in the small pipe line is ignored. This can be done by an iterative procedure which establishes a value for P_2, the pilot chamber pressure.

4.9 In this work, first of all, the estimated pressure for $P_2^{(0)}$ is assigned to be equal to P_{IN}, the inlet pressure of valve.

$$P_2^{(0)} = P_{IN}$$

If the input and output flow rates calculated by using $P_2^{(0)}$ satisfies:

$$Q_{IN}(P_2^{(0)}) = Q_{OUT}(P_2^{(0)})$$

i.e. $Q_L(P_2^{(0)}) + A_p v_x = Q_2(P_2^{(0)}) + A_2 v_y$ the $P_2^{(0)}$ is considered acceptable and the estimation of P_2 is finished before the iteration starts.

Otherwise an initial range of values of P_2 is established and an initial value postulated. Using this value a flow error test is carried out and then the value is altered by the half interval iterative technique given as below.

4.10 The iterative process commences by estimating a value for pressure P_2 using the equation:

$$P_2^{(k)} = \frac{P_{2H} + P_{2L}}{2} \qquad \ldots\ldots (4.10)$$

where

$P_{2H} = P_{IN}$. the inlet pressure

and $P_{2L} = P_3$. the drain line pressure

k - the number of iteration (initial value is 1)

i) The pressure $P_2^{(k)}$ obtained from above equation is used to calculate the input and output flow rates of pilot stage. i.e.

$$Q_{IN} = Q_L + Q_{VX}$$

$$= K_{or} \sqrt{(P_{IN} - P_2^{(k)})} + A_p \, v_x \qquad \ldots\ldots (4.11)$$

$$Q_{OUT} = Q_2 + A_2 \, v_y$$

$$= \pi \, C_d \, D_2 \, Y \, Sin(\alpha_2) \sqrt{2(P_2^{(k)} - P_3)/\rho} + A_2 \, v_y \qquad \ldots\ldots (4.12)$$

ii) flow and pressure errors are estimated by:

a) flow error(net flow): $\Delta Q = Q_{in} - Q_{out}$

b) pressure error: $\Delta P = P_2^{(k)} - P_2^{(k-1)}$

iii) Halving iterative interval for the estimation of value of P_2, if necessary.

a) If the flow error between the flows is within a set tolerance Δf, i.e.

$$| \Delta Q | \leqslant \Delta f$$

the pressure value chosen $P_2^{(k)}$ is considered acceptable, i.e.

$$P_2 = P_2^{(k)}$$

and then the iteration for pressure P_2 at this time point is finished. Figure 4.3(a) shows this case.

b) If $|\Delta Q| > \Delta f$ but $|\Delta P| \leqslant \Delta p$, as shown in figure 4.3(b), it shows that although the flow error is outside the flow tolerance Δf the estimated pressure $P_2^{(k)}$ has been near the real value P_2, i.e. pressure error obtained has been within a set tolerance Δp. In this case the estimated pressure value $P_2^{(k)}$ may be considered acceptable, i.e.

$$P_2 = P_2^{(k)}$$

and then iterative process is finished.

c) Otherwise the calculations are repeated using a new halved iterative interval $[P_{2L}, P_{2H}]$.

case (a) :

$$P_{2L} = P_2^{(k)} \text{ and } P_{2H} = P_{2H} \text{ for } Q_{IN} > Q_{OUT} \quad \ldots\ldots (4.13)$$

This is because the estimated value $P_2^{(k)}$ is still smaller than the real value P_2. This case is shown in figure 4.4(a).

or **case (b)** :

$$P_{2L} = P_{2L} \text{ and } P_{2H} = P_2^{(k)} \text{ for } Q_{IN} < Q_{OUT} \quad \ldots\ldots (4.14)$$

This is because the estimated value $P_2^{(k)}$ has been larger than the real

value P_2. This case is shown in figure 4.4(b).

The iteration process continues until an acceptable value for P_2 is obtained. The flow chart of this iterative procedure is shown in figure 4.5.

4.11 **Determination of iterative tolerances.** In order to compute the pilot chamber pressure by iteration, flow and pressure error tests are required. The error tolerances employed can be specified by the user or calculated in terms of maximum pressure and flow rate values supplied by the user using tolerance proportional functions. Suitable tolerance functions are:

i) Flow rate tolerance:

$$\Delta f = Q_{max}/100000$$

ii) Pressure tolerance:

$$\Delta p = P_{2max}/1000$$

4.12 The two stage relief valve has been modelled using this procedure, and the model has been incorporated to the HASP model library with the mnemonic PCA1. A full description and documentation are given in Appendix B of this thesis.

Test on Relief Valve Model PCA1

4.13 **Test Circuit.** To check the behaviour of the model PCA1, it was incorporated into a digital simulation for the dynamic response of a two stage relief valve. Figure 4.6 is a schematic of a hydraulic circuit used to obtain the dynamic response of a two stage relief valve. The prime mover of the test system supplies a constant speed (1500 rev/min) to the pump. The hydraulic pump delivers flow (40 l/min) against the relief valve under test. When the DCV is switched off the flow is

directed against the relief valve. The base pressure may be adjusted by an orifice in the directional control valve (In this simulation, the base pressure is chosen as 30 bar).

4.14 The computer block diagram necessary to produce a HASP simulation of the circuit is shown in figure 4.7. The model of the two stage relief valve is labelled PCA1 and has two external links and three internal links. Three internal links are designed to display the displacements of the main and pilot poppet valves and the pilot pressure P_2. The simulation data is shown in tables 4.1 to 4.2.

4.15 **Simulation Results.** Figure 4.8 shows the position of 2 port manual directional control valve and figure 4.9 shows the flow rate delivered from pump. The inlet pressure of relief valve has an initial value of 30 bar (base pressure) when the DCV is in the open position. After 0.01 second, the DCV is switched off and the flow is directly against the two stage relief valve. The dynamic responses of pressures and displacements are shown in figures 4.10 and 4.11 respectively. The flow rate through DCV is shown in figure 4.12 and the flow rate relieved from the two stage relief valve is shown in figure 4.13. Because the fluid compressibility is negligible due to the small volume of the pilot stage, the pressure of pilot stage will be equal to the system pressure (inlet pressure of valve) when the pilot poppet is closed (at time 17 ms to 18.5 ms) and the results in figure 4.14 have shown this.

4.16 In order to check the validity of the iterative process a fully dynamic model was developed by the author and given the model PCR1. It also has two external, three internal links and is described in the Appendix C of this thesis. This model was simulated to the same test as PCA1. The results obtained using the model PCR1 which incorporates compliance are shown in figure 4.15 and have good agreement with those obtained using PCA1. A comparison of the simulation speed for both test systems is given by the table 4.A.

Table 4.A: Comparison of simulation speed

Working Condition	Algorithm	CPU time (second)
for PCR1 system	KUTMER	221.2
for PCA1 system	KUTMER	188.4
Ttotal = 0.1 sec.		
Tprint = 0.001 sec.		

ITERATION TO COMPUTE PRESSURE FOR A PIPE WITH SMALL VOLUME

Introduction

4.17 Supposing we have a pipe in a system whose compliance is considered negligible, it would be useful to apply the method of iteration to compute its pressure, rather than by integration. For instance, consider a simple pump controlled actuator circuit shown in figure 4.16. The fluid compliance in the pipe which is linked with the pump is very low due to its small fluid volume. i.e.

$$\frac{dP_1}{dt} >> \frac{dP_2}{dt} \qquad \ldots\ldots (4.15)$$

If an iterative procedure can be used to compute the pressure in the shorter pipe, the stiffness problem can be avoided.

Structure of Iterative Subroutine

4.18 **Physical Structure.** According to the HASP linking rules described in Chapter 2 the non-memory algebraic models, pump PU00 and orifice OR00, can not be directly linked together, and they also can not be linked with a model in which only an iterative procedure of pressure exists. Otherwise the system model linking will fail due to the linking difficulty between two algebraic models, which has been discussed in chapter 2.

4.19 In order to overcome this kind of linking difficulty caused by the implicit

relationship between non-memory algebraic models, the pressure P_1 in the iterative subroutine must be set to a pseudo-state variable whose derivative is zero as described in Chapter 2. In this case, since the pressure P_1 is a state variable on the external links of the model, this iterative subroutine becomes a memory instantaneous model, and hence it can be linked with adjacent non-memory algebraic models PU00 and OR00 etc. without any linking difficulty. The basic structure of the iterative subroutine therefore must include the expressions:

a) statement $\frac{dP_1}{dt} = 0$, (for breaking possible linking problem only)

b) an iterative procedure to determine the real value of P_1

Since P_1 is a state variable, the initial condition of pressure is specified by the user, or by default as zero. However the pressure P_1 at $t > 0$ is computed by an iterative procedure and will not be changed by the integrator.

4.20 **Computer Model Structure.** This iterative **subroutine** is designed to compute pressure and link two algebraic models. Since the structure of the iterative **procedure** to determine the real value of P_1 would change according to different adjacent component models, this iterative **subroutine** is modelled as a modular component model (PIA5) whose computer model structure is shown in figure 4.17.

4.21 In the main section of model PIA5, the pressure value P is set to a pseudo-state variable in order to avoid linking problems. The modular subroutine ITSEL is called to select different iterative procedure subroutine, which is written by the program generator according to the number N of using the same model PIA5 in same circuit being investigated.

4.22 The modular iterative procedure subroutine is written by the program generator in terms of relevant pipe iterative model, for instance, the iterative procedure subroutine ITER1 is written for PIA501, ITER2 for PIA502, ..., ITER10 for PIA510. The iterative process of *ith* pressure is completed in the subroutine ITERi. The number i, flow and pressure error tolerances are input, and the pressure obtained is output from entry arguments of the modular subroutine. The

explanation for modular subroutines ITSEL and ITERi is given in the simulation test of model PIA5 described at next section. The modular pipe model PIA5 has been developed in the HASP model library and a full description and documentation are given in Appendix B of this thesis.

Iterative Procedure for Pipe Pressure

4.23 **Introduction.** In an actual search process, the known pipe pressure $P(t_{n-1})$ at last time point t_{n-1} is used to start the iteration calculation - which will then be very rapid. According to the iterative interval determined by a procedure to be shown below, the pipe pressure $P(t_n)$ at current time point t_n is computed by a half-interval iterative technique discussed earlier.

4.24 **Determine Iterative Interval.** The pressure value at current time point t_n is usually either larger or less than that at last time point t_{n-1}. Thus the pressure value at last time point can be used to be upper or lower bound of iterative interval for determining pressure value at current time point. The another boundary value of iterative interval can be found by means of following procedures:

i) **Determine a search direction.** Using a pressure value $P(t_{n-1})$ at last time point, the flow error ΔQ (algebraic sum of flow rates from and to pipe) is obtained by calling model subroutines connected to this pipe model.

4.25 In a special case, the flow error value equals zero, i.e. the total input flow rates equal total output flow rates, and hence the pressure value $P(t_n)$ at current time point t_n will be equal to previous value $P(t_{n-1})$. In this case, the search stops and returns back to main section of model program. It can be expressed by:

$$P = P(t_n) = P(t_{n-1}), \quad \text{if } \Delta Q = 0.$$

4.26 In most cases, the flow error is not equal to zero, the search direction of another bound value of iterative interval must be determined below.

4.27 If the flow error ΔQ is larger than zero, i.e. the total input flow rates are larger than total output flow rates. In this case, the real pressure value $P(t_n)$ should be larger than the value of $P(t_{n-1})$, therefore the search direction is defined to be positive and the step of search is assumed to be 20% of pressure $P(t_{n-1})$ because usually no large change of pressure value occurs between two time points. Thus the coefficient of search direction is set to 0.2, i.e.

$$K = 0.2, \quad \text{if } \Delta Q > 0,$$

4.28 Similarly, if the flow error ΔQ is less than zero, i.e. the total input flow rates are less than total output flow rates, the real pressure value $P(t_n)$ should be less than $P(t_{n-1})$. So the search direction is defined to be negative. In this case, the coefficient of search direction is set to -0.2, i.e.

$$K = -0.2, \quad \text{if } \Delta Q < 0,$$

ii) **Find an initial iterative interval** $[P_{LOW}, P_{UP}]$

a) $K = 0.2$ (positive search direction)

In this case, the lower bound P_{LOW} of iterative interval is

$$P_{LOW} = P(t_{n-1})$$

The upper bound P_{UP} of iterative interval is determined below.

$$P^{(k)} = P^{(k-1)} + K\ P^{(k-1)}$$

If the flow error $\Delta Q(P^{(k)}) \leqslant 0$, the real pressure $P(t_n)$ must be existed in the interval $[P_{LOW}, P^{(k)}]$, in this case,

$$P_{UP} = P^{(k)}$$

Otherwise let k=k+1 repeat above calculation until the flow error is less than zero. Here initial $P^{(k-1)}$ is $P(t_{n-1})$ and the final iterative interval is $[P(t_{n-1}), P^{(k)}]$.

b) $K = -0.2$ (negative search direction)

In this case, the upper bound P_{UP} of iterative interval is

$$P_{UP} = P(t_{n-1})$$

The lower bound P_{LOW} of iterative interval is determined below.

$$P^{(k)} = P^{(k-1)} + K\ P^{(k-1)}$$

If the flow error $\Delta Q(P^{(k)}) \geqslant 0$, the real pressure $P(t_n)$ must be existed in the interval $[P^{(k)}, P_{UP}]$, in this case,

$$P_{LOW} = P^{(k)}$$

Otherwise let k=k+1 repeat above calculation until the flow error is equal to or larger than zero. Here initial $P^{(k-1)}$ is $P(t_{n-1})$ and the final iterative interval is $[P^{(k)}, P(t_{n-1})]$.

4.29 A logical block diagram describing above automatic search

procedure of an initial iterative interval is shown in figure 4.18. Once an initial iterative interval $[P_{LOW}, P_{UP}]$ is found, the procedure of finding initial iterative interval is finished and the execution of the program will automatically go to the half-interval iterative section to carry on the search of the real pressure value $P(t_n)$.

4.30 **Iteration to Compute Pipe Pressure.** After obtaining an iterative interval from above automatic procedure, the iterative process commences by estimating a value for pipe pressure using the equation:

$$P^{(k)} = \frac{P_H + P_L}{2} \qquad \ldots\ldots (4.16)$$

where $P_H = P_{UP}$ and $P_L = P_{LOW}$

where k is the number of iteration (initially k=1).

and then a real pressure value P for pipe can be found in the interval $[P_H, P_L]$ by following stages:

i) The pressure $P^{(k)}$ obtained from above equation is used to calculate the flow error (algebraic sum of input and output flow rates of the pipe) by calling adjacent models.

ii) flow and pressure errors are estimated by:

a) flow error: $\Delta Q = \sum Q$

b) pressure error: $\Delta P = P^{(k)} - P^{(k-1)}$

iii) Halving iterative interval for the estimation of value of P, if

necessary.

a) If the flow error between the flows is within a set tolerance Δf, i.e.

$$| \Delta Q | \leqslant \Delta f$$

the pressure value chosen $P^{(k)}$ is considered acceptable, i.e.

$$P = P^{(k)}$$

and the iteration for pressure P at this time point is finished.

b) If $| \Delta Q | > \Delta f$ but $| \Delta P | \leqslant \Delta p$, (similar with that shown in figure 4.3(b)), it shows that although the flow error is outside the flow tolerance Δf the pressure chosen has been near the real value for P, i.e. pressure error obtained has been within a set tolerance Δp. In this case the estimated pressure value $P^{(k)}$ may be considered acceptable, i.e.

$$P = P^{(k)}$$

and then iterative process is finished.

c) Otherwise the calculations are repeated using a new halved iterative interval $[P_L, P_H]$shown below, when

$$P_L = P^{(k)} \text{ and } \quad P_H = P_H \quad \text{for } \Delta Q > 0 \qquad \ldots\ldots (4.17)$$

or

$$P_L = P_L \text{ and } P_H = P^{(k)} \text{ for } \Delta Q < 0 \quad \ldots\ldots (4.18)$$

The iteration process continues until an acceptable value for P is obtained.

4.31 **Determination of Iterative Tolerances.** In order to compute the pipe pressure by iteration, flow and pressure error tests are required. The error tolerances employed can be specified by the user or calculated in terms of maximum pressure and flow rate values supplied by the user using tolerance proportional functions. Suitable tolerance functions are:

Flow rate tolerance:

$$\Delta f = Q_{max}/100000$$

Pressure tolerance:

$$\Delta p = P_{max}/1000$$

Test on Pipe Instantaneous Model PIA5

4.32 The purpose of this simulation test is to investigate the operating reliability of the model (PIA5), for instance, the reliability of the iterative procedure when the flow rate through a pipe changes its direction, and the pressure characteristic under the cavitation, etc.

4.33 **Case 1:** In a pipe orifice system with a directional control valve, which is shown in figure 4.19, the direction of the flow rate through two orifices can be changed by a two position four port directional control valve. The fluid compressibility in the pipe linked with two orifices is assumed to be negligible due to a small volume, hence the pressure characteristic in the pipe can be described by an instantaneous model (PIA5).

4.34 **Component model details for test.** The simulation program for the testing system was produced using the HASP program generator. In order to perform the program generation, the circuit data details in table 4.B were firstly obtained from the computer model linking diagram shown in figure 4.20. The hydraulic component models are indicated on this diagram.

Table 4.B: Component model details for the PIA5 test

12
PM0001 12
PU0001 08 09 12
PI0501 09 10 02
PC0101 10 11
DE0101 01
DC0101 01 02 03 04 05
TK0301 05 08 11
PI0502 03 06
OR0001 06 13
PIA501 13 14
OR0002 14 07
PI0503 07 04

In order to operate the system simulation, the main pump prime mover speed must be supplied by the user and it is transmitted to the pump model as a flow variable on link 12. The system pressure depends on the cracking pressure Pc of the relief valve and it is the effort variable on links 9,10 and 2. The operating position of the directional control valve is available as the effort variable on link 1. The pressure and flow rate for the testing model PIA5 are respectively the effort and flow variables on links 13 and 14.

Test data. The initial pressure of the system was set to 16 bar and the initial operating position of the directional control valve is closed one. The test data used in the simulation is shown in table 4.3.

4.35 **Simulation results and discussions.** In order to investigate the reliability of the model PIA5 when the flow rate through a pipe changes its direction, an operating characteristic of a directional control valve with 4 position 3 port is chosen and shown in figure 4.21. In this case, the spool valve operates at central position as $t < 0.2$ sec. and $t > 0.815$ sec., otherwise it will operate at the operating positions, i.e. upper or lower position, and the valve will keep fully open in different ways.

4.36 **Flow characteristic.** When the directional control valve works at the central position, the flow rate through the valve is zero. If the spool moves to upper position from central position, the flow rate from pump goes pass the directional control valve to the pipe No.2. If the spool moves to lower position, the flow rate will go to the pipe No.3 through the directional control valve. According to the operating characteristic of the valve chosen earlier, the direction of the flow rate through the pipe iterative model PIA5 is changed once and the flow characteristic obtained is shown in figure 4.22.

4.37 **Pressure characteristic of pipe between two orifices.** The pressure value computed by the model PIA5 is shown in figure 4.23. At the beginning, the pressure has an initial value of 16 bar when the spool valve is in the central position and jumps to a peak value of 26 bar when the spool is moved, and then goes down to a steady value about 19 bar due to the fluid compressibility effect of pipe PI0501. The pressure value jumps again to another peak of 21.5 bar as the flow rate through the pipe changes its direction, and then it goes back to the stable value again. Because the flow exists in the pipe chamber and the pressure value is higher than before, the second pressure peak is lower than first one as the flow direction of the pipe described by model PIA5 is changed. When the spool valve moves back the closed position, an equivalent step change of the flow rate through the PIA5 occurs, so the pressure value increases a small amount and then stays constant.

4.38 **Comparison of simulation results using PIA5 and PI05.** A comparison test was done by replacing PIA5 using PI05 accounting for pipe compressibility. The comparison of the simulation results for both systems is shown in figure 4.24. Using the model PI05, the pressure peak value for the response of a step input flow rate is lower than one of using PIA5 model because the fluid compressibility is taken into account in PI05. This is the only difference between the two systems. Moreover a comparison of the simulation speed for both systems is given by the table 4.C.

Table 4.C: Comparison of simulation speed

Working Condition	Algorithm	Step Length (second)	CPU time (second)
for PI05 system	RK4	$1x10^{-5}$	1810
		$> 1x10^{-5}$	(failure)
for PIA5 system Ttotal = 6 sec. Tprint = 0.01 sec. L = 0.2 m	RK4	$2x10^{-4}$	210

where L is the length of the pipe connected to two orifices.

4.39 **Case 2:** In order to further examine the reliability of the instantaneous pipe model PIA5, consider a hypothetical multi-pipe-orifice system shown in figure 4.25. Supposing ten pipes connected with two orifices have very small fluid volume, the fluid compressibility in these pipes can be negligible, and hence these pipes can be described by the instantaneous model PIA5. A computer linking diagram for this system is shown in figure 4.26.

4.40 **Test data.** The test data used in the simulation is shown in table 4.4. and the simulation is carried out using the RK4 integration routine.

4.41 **Simulation results and analysis.** As a test of the system the pump was assumed to be initially at rest, and at time $t = 0$ started, to deliver instantaneously its full flow, as shown in figure 4.27. Figure 4.28 shows the pump delivery pressure (first pipe) and the pressure in the second pipe. The pressure in the model PI0501 of the first pipe is a dynamic value and supplied by the integrator. The pressure in the model PIA501 (second pipe) is computed by an iterative procedure coded in the subroutine ITER1. Pressure difference between two values is the pressure drop across the first orifice. The initial pressure values in both pipes are set to 16 bar. The flow rate through the first orifice is shown in figure 4.29.

4.42 Figure 4.30 shows the pressure responses in the pipes No.10 and No.11. Since the initial pressure values in pipes are set to 16 bar and the tank pressure is zero bar, at the beginning of the simulation, the pressures in pipes go down. When the more flow rate comes in the pipe, the pipe pressure values are built up until their stable value are obtained. The pressures obtained here are instantaneous value and computed by the iterative procedures coded in the subroutines ITER9 and ITER10 respectively. The flow rate through the pipes No.10 and No.11 is shown in figure 4.31.

4.43 The results obtained from above two reliability tests show that the pipe instantaneous model works very well even when the flow rate through the orifices changes its direction or the same model PIA5 is employed more than once at a same circuit.

4.44 **Programming of Iterative Procedure Subroutines.** According to the component models linked to the model PIA5, the HASP program generator will automatically write out the iterative procedure subroutine for i **th** short pipe by means of the iterative procedure described in paragraph 4.23 to 4.30.

i) Subroutine ITSEL. Since the model PIA5 is employed ten times in this simulation, the subroutine ITSEL will first be written automatically below by the HASP program generator.

```
C
C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C %%%% SUBROUTINE ITSEL - SELECTING ITERATIVE SUBROUTINES
C %%%% GENERATED BY HASP PROGRAM GENERATOR
C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE ITSEL(N,YM,DEQ,DEP,LIMIT)
      IF(N.EQ.1)CALL ITER1(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.2)CALL ITER2(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.3)CALL ITER3(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.4)CALL ITER4(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.5)CALL ITER5(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.6)CALL ITER6(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.7)CALL ITER7(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.8)CALL ITER8(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.9)CALL ITER9(YM,DEQ,DEP,LIMIT)
      IF(N.EQ.10)CALL ITER10(YM,DEQ,DEP,LIMIT)
      RETURN
      END
```

4.45 **Explanation for arguments, variables of model ITSEL.** The iterative error tolerances, initial or default value of pressure and the number of using PIA5 are supplied from the main section of PIA5 (see figure 4.17), they are respectively expressed as **DEQ**, **DEP**, **YM** and **N. LIMIT** is an indicator of integration status and is described in detailed in Chapter 6. After the ith pressure variable P_i is computed by calling the ith subroutine ITERi, P value will be stored in sign **YM** and output to main section of PIA5.

4.46 The first line of the model ITSEL computes the pressure of model PIA501, it is expressed as:

```
      IF(N.EQ.1)CALL ITER1(YM,DEQ,DEP,LIMIT)
```

The tenth line computes the pressure of PIA510 due to N=10, i.e.

```
      IF(N.EQ.10)CALL ITER10(YM,DEQ,DEP,LIMIT)
```

When the i **th** PIA5 is called in the simulation, N=i, and then the i **th** iterative procedure subroutine is selected to compute the i **th** pipe pressure.

4.47 **Subroutine ITER1.** In this simulation, the model PIA5 is employed ten times, and hence the ten subroutines ITER1, ITER2, ..., and ITER10 are written out respectively. Here only the coding of ITER1 is listed below.

Subroutine ITER1

```
C
C %%%%
```

```
C
C %%%% TO FIND AN INITIAL ITERATIVE INTERVAL FOR AN ITERATIVE LOOP
C %%%% I.E. [Y1,Y2]
C
C %%%% SET INITIAL ESTIMATED PRESSURE PL(I)
C
      I=1
      PL(I)=YM
      IF(PL(I).EQ.0.D0)PL(I)=0.5D0
C
C %%%% CALCULATE FLOW ERROR USING INITIAL ESTIMATED PRESSURE PL(I)
C
      CALL OR00(EFF(4),PL(I),FLO(4),FLO(5),LIMIT,CON4,ICON4
     +  )
      CALL OR00(PL(I),EFF(7),FLO(6),FLO(7),LIMIT,CON6,ICON6
     +  )
      QSUM=FLO( 5)+FLO( 6)
C
C %%%% FLOW ERROR TEST FOR INITIAL ESTIMATED PRESSURE PL(I)
C
      IF(DABS(QSUM).LE.DEQ)THEN
        YM=PL(I)
        RETURN
      END IF
C
C %%%% DETERMINE SEARCH DIRECTION COEFFICIENT FOR FINDING INITIAL
C %%%% ITERATIVE INTERVAL
C
      IF(QSUM.LT.-DEQ)THEN
        CK=-0.2D0
      ELSE
        CK=0.2D0
      END IF
C
C %%%% FIND AN REASONABLE INITIAL ITERATIVE INTERVAL [Y1,Y2]
C
 999  PL(I+1)=PL(I)+CK*PL(I)
      IF(DABS(PL(I+1)-PL(I)).LE.DEP)THEN
        Y1=-1.D0
```

```
        Y2=100*PL(I)
        GOTO 888
       END IF
       CALL OR00(EFF(4),PL(I+1),FLO(4),FLO(5),LIMIT,CON4,ICON4
     +  )
       CALL OR00(PL(I+1),EFF(7),FLO(6),FLO(7),LIMIT,CON6,ICON6
     +  )
       QSUM=FLO( 5)+FLO( 6)
       IF(DABS(QSUM).LE.DEQ)THEN
        YM=PL(I+1)
        RETURN
       END IF
       IF(CK.GT.0.D0)THEN
        IF(QSUM.GT.DEQ)THEN
         I=I+1
         GOTO 999
        ELSE
         Y1=PL(I)
         Y2=PL(I+1)
        END IF
       GOTO 888
       ELSE
        IF(QSUM.LT.-DEQ)THEN
         I=I+1
         GOTO 999
        ELSE
         Y1=PL(I+1)
         Y2=PL(I)
        END IF
       END IF
 888   CONTINUE
C
C %%%% START ITERATIVE LOOP:
C %%%% SET APPROPRIATE PREDICTED PRESSURE VALUE Y3
C %%%% CALL MODEL SUBROUTINES LINKED TO THE PIPE ITERATIVE
C %%%% MODEL TO FIND RELATED PREDICTED FLOWRATE
C
 10    YM=(Y1+Y2)/2.D0
       CALL OR00(EFF(4),YM,FLO(4),FLO(5),LIMIT,CON4,ICON4)
```

```
      CALL OR00(YM,EFF(7),FLO(6),FLO(7),LIMIT,CON6,ICON6)
C
C %%%% DQ - ALL FLOWS IN AND OUT OF PIPE ITERATIVE MODEL
C
      QSUM=FLO( 5)+FLO( 6)
      DQ=QSUM
C
C %%%% LOGICAL COMPARISON SECTION
C
      IF(DABS(DQ)-DEQ)100,20,20
 20   IF(DQ)30,50,50
 30   IF(DABS(YM-Y1)-DEP)100,40,40
 40   Y2=YM
      GOTO 10
 50   IF(DABS(YM-Y2)-DEP)100,60,60
 60   Y1=YM
      GOTO 10
 100  RETURN
      END
```

A modification of the HASP program generator for writing out subroutines ITSEL and ITERi has been described in Appendix A of this thesis.

An Example for the Application of the Instantaneous Pipe Model to the Simulation of a Simple Hydrostatic Transmission with an Orifice in the Main Loop

4.48 In a simple hydrostatic transmission with an orifice shown in figure 4.32, the orifice simulates pressure drop in long pipe downstream of relief valve. The fluid compressibility in the pipe connected to the main pump, relief valve and orifice is assumed to be negligible. Hence the behaviour of this pipe can be described by an instantaneous model PIA5. The computer linking diagram is shown in figure 4.33.

4.49 **Simulation Results and Discussions.** As a test of the system the pump was assumed to be initially at rest, and at time $t = 0$ started, to deliver instantaneously its full flow, as shown in figure 4.34, which falls from its peak value of 1.25 l/s as the pressure rises. Figures 4.35 and 4.36 show the pump delivery pressure and the relief valve flow. It can be seen that the relief valve opened after about 0.2 ms, and closed again after 0.9 ms, thus reducing the initial peak value of pressure. The relief valve model was an instantaneous in action - in practice it is unlikely that a valve would react so quickly to a transient. The subsequent behaviour is shown in figures 4.37 and 4.38. The motor started to move at about 0.2 ms, when its static friction was overcome. The resultant oscillations in motor speed are shown and are considerably lower than would have been the case if pipe friction had been presented. Test data used in the simulation is shown in table 4.5.

The simulation results obtained by using the model PIA5 are the same with those obtained by using the dynamic model PI05 when the pipe volume is small (e.g. we chose pipe l(length) = 0.4 m, d(diameter) = 15 mm). The simulation speed is however faster using PIA5 as compared with for PI05.

ITERATIVE TECHNIQUE APPLIED TO THE SIMULATION OF FLOW CONTROL SYSTEMS

Introduction

4.50 A particular simulation difficulty occurs in meter-in and meter-out flow control systems. Often with these systems the control orifice is often placed extremely close to the actuator. At the end of the stroke the volume between the piston face and the orifice will be extremely small and if normal integration techniques were used to compute the pressure in the piston face, an impracticably small step length would be required since

$$\frac{dP}{dt} \rightarrow \infty$$

4.51 This problem has often been encountered in the HASP system where the volume between the actuator and the orifice is automatically varied during the stroke length.

4.52 One obvious way to avoid the difficulty is to iterate to obtain the pressure as in the model PIA5 referred to above. However, when the actuator piston moves a significant distance from its end the compressibility of the oil in the actuator will be important in assessing the dynamics of the system.

The description below gives a technique for using iteration when the volume is low and integration when the compliance of the oil becomes significant.

4.53 In the HASP system the fluid in the actuator is lumped with that of the adjacent pipe, and hence the equation of that adjacent pipe becomes

$$\frac{dP}{dt} = \frac{\beta}{V_p + V_{IN}} (Q_{IN} - A\ v) \qquad \ldots\ldots (4.19)$$

where P - pipe pressure in bar
V_p - pipe volume in litre
V_{IN} - actuator piston chamber volume in litre
Q_{IN} - input flow rate to piston chamber in l/s
A - piston area in m^2
v - actuator velocity in m/s
β - effective fluid bulk modulus in bar

When $V_p + V_{IN}$ becomes a small value, it is desirable to ignore the equation and iterate to obtain P, as described above.

4.54 However this leads to computational difficulties as it would mean the removal of an equation from the integration matrix. Instead of this the equation was left intact in the simulation, but when the values become less than the critical value V_c, $\frac{dP}{dt}$ is made zero and the value of P overwritten by the value obtained from iteration.

4.55 In this model the cavitation is allowed when the working pressure is below the saturation pressure, i.e.

if $P < P_{sat}$, entering cavitation

The minimum value of operating pressure can not be lower than -1 bar gauge. When the pressure approaches this condition, cavitation will occur and the fluid will be vapourised. This condition can be predicted from Henry's law[15][18]. In the model, the pressure can not be limited to -1 bar gauge as this would result in an infinite value for the predicted bulk modulus for the gas (see paragraph 4.58). For this reason, the minimum pressure is limited to -0.999 bar and is taken into account using the following statement.

if $P < -1$ bar, $P = -0.999$ bar

4.56 This combined procedure has been coded in a pipe model which has been incorporated to the HASP model library with the mnemonic PIA6. A full description and documentation are given in Appendix B of this thesis.

Procedure to Obtain Pressure

4.57 Consider a linear actuator system with a control orifice shown in figure 4.39. It is assumed that the pipe volume V_p is very small comparing with 10% of total actuator volume. The fluid volume affecting the dynamics of a pipe connected with an actuator is varied and depends on the movement of the actuator piston. Before the actuator piston extends, this fluid volume is only the pipe volume. After the piston extends, the effective fluid volume should include the pipe volume and the actuator piston chamber volume. Therefore the pressure value in this pipe can be obtained either in dynamic region by integration or in instantaneous region by iteration and it will depend on how large the actuator piston chamber volume V_{IN} is. In order to judge when the iteration or integration algorithm is used, a critical volume V_c is defined as follows:

$$V_c = 10\% \, V_{A\,max}$$

or $$X_A = 10\% \, X_{A\,max}$$

where X_A - displacement of actuator piston

$X_{A\,max}$ - maximum displacement of actuator piston

$V_{A\,max}$ - maximum volume of actuator piston chamber

Two operating regions are defined:

$V_{IN} > V_c$ - the dynamic region in which the integration is used.

and $V_{IN} \leqslant V_c$ - the instantaneous region in which the iteration is used.

4.58 **Obtain Pressure Value in Dynamic Region.** If the fluid volume of the actuator piston chamber is larger than the setting critical volume V_c, the fluid compressibility is taken into account and the pressure behaviour of the pipe is described by the equation (4.19). i.e.

$$\frac{dP}{dt} = \frac{\beta}{V_p + V_{IN}} (Q_{IN} - A\ v)$$

Where the pressure P is a state variable and is supplied by the integrator by means of the derivative of pressure. β is an effective bulk modulus and is computed by

$$\beta = \beta_f \qquad \text{if } P > P_{sat}$$

$$\beta = \frac{1}{\frac{1}{\beta_f} + \frac{d_0 (P_{sat} - P)}{n (P+1)^2}} \qquad \text{if } P \leqslant P_{sat} \quad \text{(entering cavitation)}$$

Hence in the dynamic region, the pressure transient behaviour is dependent with the fluid compressibility, and the mathematical model allows the cavitation by assuming an air release mechanism[18] as shown in above equation.

4.59 **Obtain Pressure Value in Instantaneous Region.** If the fluid volume of the actuator piston chamber is equal to or less than the critical volume V_c, the dynamic effect of the fluid compressibility is ignored and the pipe pressure value is computed by iteration. The whole practical process to obtain this pressure value is identical with that in pipe instantaneous model PIA5, which was given in paragraphs 4.23 to 4.30.

4.60 In the instantaneous region cavitation is allowed when the working pressure is lower than the saturation pressure and the minimum working pressure is set to -0.999 bar, i.e.

$P < P_{sat}$ entering cavitation

if $P < -1$ bar $P = -0.999$ bar

In this region the pressure P is set to a pseudo-state variable and $\frac{dP}{dt} = 0$ is used in order to avoid linking difficulty.

Test on the Pipe Model PIA6

4.61 For a pipe combined model PIA6, the first characteristic we want to investigate is the transition between iterative and integration procedure sections, and the second one is the pressure characteristic under the cavitation. As a simulation example for a reliability test, a linear actuator system with a control orifice shown in figure 4.39 is considered, and then a schematic of the operation region for the iteration and integration and a corresponding computer model linking diagram of the system are given in figures 4.40 and 4.41 respectively.

4.62 **Component Model Details for Test System.** The circuit data details in table 4.D of performing the program generation using the HASP program generator are obtained from the computer model linking diagram of the system shown in figure 4.41. The hydraulic component models are indicated on this diagram.

Table 4.D: Component model details for the PIA6 test

12
PM0001 12
PU0001 08 09 12
PI0501 09 10 02
PC0101 10 11
DE0001 01
DC0101 01 02 03 04 05

TK0301 05 08 11
PI0502 03 06
OR0001 06 13
PIA601 13 14 01
AL0101 14 07 01 02
PI0601 07 04 02

The prime mover of the test system supplies a constant speed to the pump as a flow variable on link 12. The main pump delivery flow and pressure are respectively the flow and effort variables on link 9. The system pressure is the effort variable on links 9, 10 and 2. The flow rate discharged from a relief valve is available as the flow variable on links 10 and 11. The net flow rate through the directional control valve is supplied as the flow variable on link 2. The operating position of the directional control valve is available on link 1. The flow rate and pressure to the piston and rod sides of the actuator are respectively supplied as the flow and effort variables on links 14 and 7. Volume changes for both sides of the actuator are transmitted to the pipe models connected to the actuator and are available on signal links 1 and 2.

4.63 **Simulation Data and Simulation Region.** The simulation was carried out using the test data shown in tables 4.6 to 4.7 and a region of changing actuator volume is chosen from zero to the double of critical volume V_c, i.e.

$$V_{IN} = 0 - 20\% \, V_{A\,max}$$

4.64 **Simulation Results and Analysis.** Operating characteristic of a directional control valve shown in figure 4.42 ensures that the simulation calculation of the pressure variable is carried out alternately in iteration or integration section of the model PIA6.

4.65 **Cavitation in different regions.** From the simulation results of the pipe (PIA6) pressure variable shown in figure 4.43, there are four places where the cavitation occurs. In iteration region ($V_{IN} \leqslant 10\% V_{max}$) the cavitation occurs twice. The first cavitation appears around t=0.3 second due to a flow step input when the actuator starts to extend, and the second one appears around t=1.2 sec. when the operating position of the directional control valve returns back to upper position again. In integration region, ($V_{IN} > 10\% V_{max}$), cavitation also occurs twice, one is around t=0.71 to t=0.74 seconds, and another is around t=1.61 to t=1.64 seconds due to the negative flow step input to PIA6 from direction control valve.

4.66 **Characteristic of displacement and velocity.** When the spool of the direction control valve moves to upper position from central position, the flow rate from the hydraulic pump is supplied to the piston side of the actuator through a pipe, and then the actuator is extended. When the direction of the flow rate through the pipe is altered manually by the directional control valve, the net pressure force acting on the actuator piston changes its direction, and then the actuator is retracted. The displacement and velocity of the actuator shown in figures 4.44 and 4.45 display this operation.

4.67 The reliability test results and analysis show that the pipe combined model PIA6 works very well no matter when the working volume of the actuator is less or large than the critical volume V_c. Cavitation happened in practical operation also can be expressed in both instantaneous and dynamic regions of model PIA6. In order to show the application of the combined model PIA6 to the simulation of the fluid power systems with mathematical stiffness, we choose the linear actuator system with a control orifice used above, and the relevant simulation and analysis are given below.

Simulation of a Linear Actuator System with a Control Orifice

4.68 The purpose of this simulation test is to obtain a complete system characteristic and to compare the simulation results obtained by different integration methods. The PIA6 used in the simulation is the model of a pipe connected to an actuator, taking into account the variable volume which occurs as

the actuator moves. If the actuator variable volume V_{IN} is less than or equal to a critical volume V_c defined by the user (e.g. 10% V_{max}), the pressure is computed by calling ITER subroutine and it is a real pressure value, but cavitation effect is taken into account. If the volume V_{IN} is larger than V_c, the pressure value is provided by the integrator and the fluid compressibility, cavitation effect are taken into account.

4.69 In this system being simulated, except PIA6, other computer models of hydraulic components were ready in the HASP model library and the component model details for the system is identical with table 4.B. Simulation is carried out using the RK4 integration subroutine. Test data used in the simulation is same with those in tables 4.6 to 4.7.

4.70 **Simulation Results and Discussions.** Figure 4.46 shows that the directional control valve is held open for a period of time extending the linear actuator to approximately half its stroke. The valve is then centred and when the system becomes steady, moved to the opposite open position thus retracting the actuator. Finally, when the actuator is at approximately quarter stroke, the valve is again centred. Figures 4.47 and 4.48 show the actuator displacement and pressure responses respectively.

4.71 The simulation results obtained by the RK4 integrator (using PIA6 model) are the same with those obtained by GEAR or KUTMER algorithm (using PI06 model). However, when this system is very "stiff", i.e. the length of the pipe is small, the CPU time spent by the former algorithm is much less than latter one. A comparison of the simulation speed is given by the table 4.E.

Table 4.E: Comparison of simulation speed

Working Condition	Algorithm	Step Length (second)	CPU time (second)
L_3 = 5.88 m	RK4	$2x10^{-4}$	475
	GEAR	$H_{max} = 10^{-2}$	1216
		$H_{min} = 2.14x\ 10^{-6}$	
Ttotal = 6 sec.	KUTMER	$H_{max} = 10^{-2}$	943
		$H_{min} = 5.45x\ 10^{-6}$	
Tprint = 0.01 sec.	RK4 (Using PIA6)	$2x10^{-4}$	402
L_3 = 0.2 m	RK4	$1x10^{-5}$	7688
		$> 1x10^{-5}$	failure
	GEAR	$H_{max} = 10^{-2}$	1783
		$H_{min} = 3.96x\ 10^{-7}$	
Ttotal = 6 sec.	KUTMER	$H_{max} = 10^{-2}$	1278
		$H_{min} = 8.23x\ 10^{-7}$	
Tprint = 0.01 sec.	RK4 (Using PIA6)	$2x10^{-4}$	544

where L_3 is the length of the pipe connected to an actuator and an orifice.

Component Type	Parameter Sign	Value	Unit
Prime Mover	ω_p	1.5×10^{3}	rev/min
Pump	D_p	5×10^{-2}	l/rev
Pipe	d	15	mm
	L	1.2144	m
	β	1.2×10^{4}	bar
	P_{sat}	0	bar
	d_0	0.1	
	P_0	$3.x10^{1}$	bar
Directional control valve	C_1	1.2172×10^{-1}	
	C_2	3.8492×10^{-1}	
Tank	P_t	0.0	bar

Table 4.1 Data values used for simulation test of model PCA1

Parameter	Sign	Value	Unit
Pilot Stage:			
cracking pressure	P_{cr}	$2x10^{2}$	bar
mass of pilot poppet	M_p	$1x10^{-2}$	kg
spring stiffness	k_p	$7.938x10^{4}$	N/m
viscous coefficient	f_p	$5.388x10^{1}$	Ns/m
diameter of poppet seat	D_p	$4x10^{-3}$	m
area of poppet seat	A_p	$1.2566x10^{-5}$	m^2
volume of pilot stage	V_p	$7.3x10^{-6}$	m^3
angle of poppet face	α_p	$2x10^{1}$	degree
Main Stage:			
Stage mass of main poppet	M_m	$2.34x10^{-2}$	kg
spring stiffness	k_m	$4.7726x10^{3}$	N/m
viscous coefficient	f_m	$7.5x10^{2}$	Ns/m
diameter of poppet seat	D_m	$1.37x10^{-2}$	m
area of poppet seat	A_m	$1.4741x10^{-4}$	m^2
volume of main stage	V_m	$2.146x10^{-4}$	m^3
angle of poppet face	α_m	$1.5x10^{1}$	degree
diameter of spring chamber	D_{ms}	$1.4x10^{-2}$	m
viscous length of poppet	L_s	$1.98x10^{-2}$	m
diameter of control orifice	d_0	$8.0x10^{-4}$	m
length of control orifice	l_0	$5.0x10^{-3}$	m

Table 4.2 Data values used for simulation of a two stage relief valve

Parameter	Sign	Value	Unit
prime Mover	ω_p	1.5×10^3	rev/min
maximum pump displacement	X_p	2.2×10^{-2}	l/rev
fraction of full displacement	k_f	1	
internal diameter for pipe No.1	d_1	2.5×10^1	mm
internal dia. for pipes No.2,3	d_{23}	1.5×10^1	mm
length of pipe No.1	L_1	3.47	m
length of pipes No.2,3	L_{23}	5.88	m
Bulk Modulus in pipe No.1	β_1	1.2×10^4	bar
Bulk Modulus in pipes No.2,3	β_{23}	$7.x10^3$	bar
air saturation pressure	P_{sat}	0	bar
proportion of dissolved air	d_0	0.1	
relief valve cracking pressure	P_c	1.1×10^2	bar
relief valve constant	k	0.5	(L/s)/bar
directional control valve			
orifice Constant	C_1	1.4556×10^{-1}	
	C_2	1.4809×10^{-1}	
	C_{v1}	1.3399×10^{-1}	
	C_{v2}	1.3596×10^{-1}	
rated pressure	P_{rate}	$1.x10^2$	bar
diameter of orifice	d_o	6.5	mm
coefficient of discharge	C_d	0.69	
flow tolerance in PIA5	Δf	$1.x10^{-6}$	$\frac{m^3}{s}$
pressure tolerance in PIA5	Δp	$1.x10^{-2}$	bar
initial pressure in PIA5	P_0	1.6×10^1	bar
tank pressure	P_t	0.0	bar

Table 4.3 Data values used for simulation test of model PIA5

Component Type	Parameter Sign	Value	Unit
Prime Mover	ω_p	1.5×10^3	rev/min
Pump	D_p	3.33×10^{-2}	l/rev
Pipe (PIA5)	Δf	1×10^{-6}	$\frac{m^3}{s}$
	Δp	1×10^{-2}	bar
	P_0	1.6×10^1	bar
Orifice	dor	6.5	mm
	C_d	0.69	
Pipe (PI05)	β	1.8×10^4	bar
	d	1.5×10^1	mm
	L	5	m
	P_0	1.6×10^1	bar
Tank	P	0	bar

Table 4.4 Simulation Data of a Multi-pipe-orifice System

Component Type	Parameter Sign	Value	Unit
Prime Mover	ω_p	1.5×10^3	rev/min
Pump	D_p	5×10^{-2}	l/rev
Relief Valve	P_c	1.1×10^2	bar
	k	5×10^{-2}	l/s /bar
Pipe (PIA5)	Δf	1×10^{-6}	m^3/s
	Δp	1×10^{-2}	bar
	P_0	0	bar
Orifice	dor	7.5	mm
	C_d	0.69	
Pipe (PI05)	β	7×10^3	bar
	d	1.5×10^1	mm
	L	3	m
	P_0	0	bar
Motor	D_m	1	l/rev
	$K_{leakage}$	2.08×10^{-11}	
	J	2	kgm^2
Tank	P	0	bar

Table 4.5 Simulation Data of a Simple Hydrostatic Transmission

Component Type	Parameter Sign	Value	Unit
Prime Mover	ω_p	1.5×10^3	rev/min
Pump	D_p	5×10^{-2}	l/rev
Pipe (PI0501)	β	1.2×10^4	bar
	d	1.5×10^1	mm
	L	5.88	m
	P_0	0	bar
	P_{sat}	0	bar
	d_0	0.1	
Relief Valve	P_c	$7. \times 10^1$	bar
	k	0.5	(L/s)/bar
Directional Control Valve	C_1	1.4556×10^{-1}	
	C_2	1.4809×10^{-1}	
	C_{v1}	1.3399×10^{-1}	
	C_{v2}	1.3596×10^{-1}	
	P_{rate}	$1. \times 10^2$	bar
Pipe (PI0502)	β	7×10^3	bar
	d	1.5×10^1	mm
	L	5.88	m
	P_0	1.6×10^1	bar
	P_{sat}	0	bar
	d_0	0.1	

Table 4.6 Data values used for simulation test of PIA6 model

Component Type	Parameter Sign	Value	Unit
Orifice Restrictor	d_o	6.5	mm
	C_d	0.69	
Actuator	D	5.08	cm
	d	2.5	cm
	M	8.7×10^2	kg
	F_s	$2. \times 10^2$	N
	F_c	$1. \times 10^2$	N
	f_v	$5. \times 10^3$	$N/(m/s)$
	X_{max}	0.61	m
	α	0.0	degree
	X_0	0.0	m
	v_0	0.0	m/s
Pipe (PIA6)	Δf	$1. \times 10^{-6}$	$\frac{m^3}{s}$
	Δp	$1. \times 10^{-2}$	bar
	V_c	$10\% \times V_{max}$	m^3
	P_0	1.6×10^1	bar
Pipe (PI06)	d	1.5×10^1	mm
	L	1.247×10^1	m
	β	$7. \times 10^3$	bar
	P_0	2.1×10^1	bar
	P_{sat}	0	bar
	d_0	0.1	
Tank	P_{tank}	0.0	bar

Table 4.7 Data values used for simulation test of PIA6 model

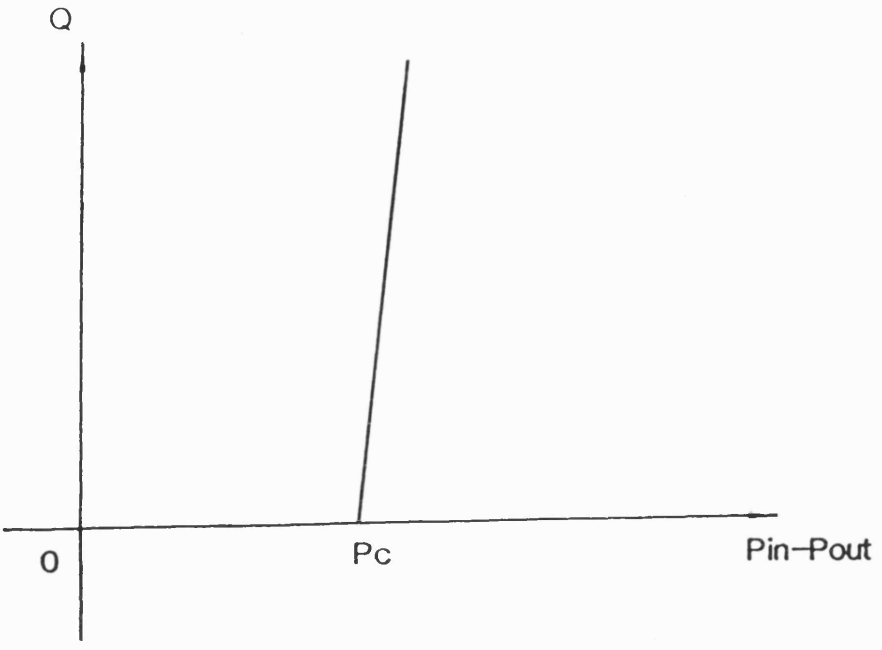


Figure 4.1 Flow pressure characteristics of relief valve

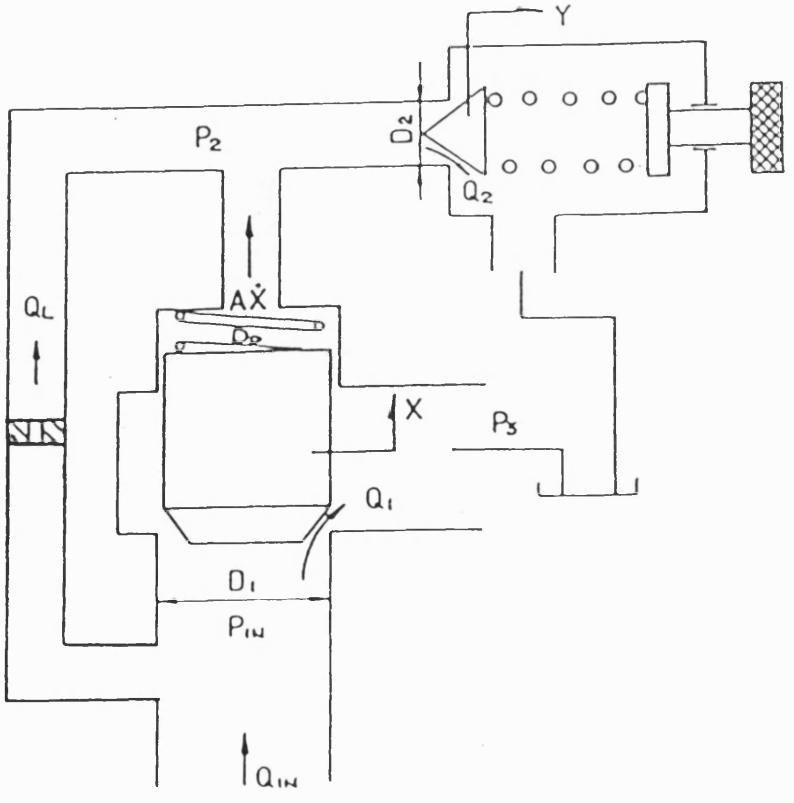
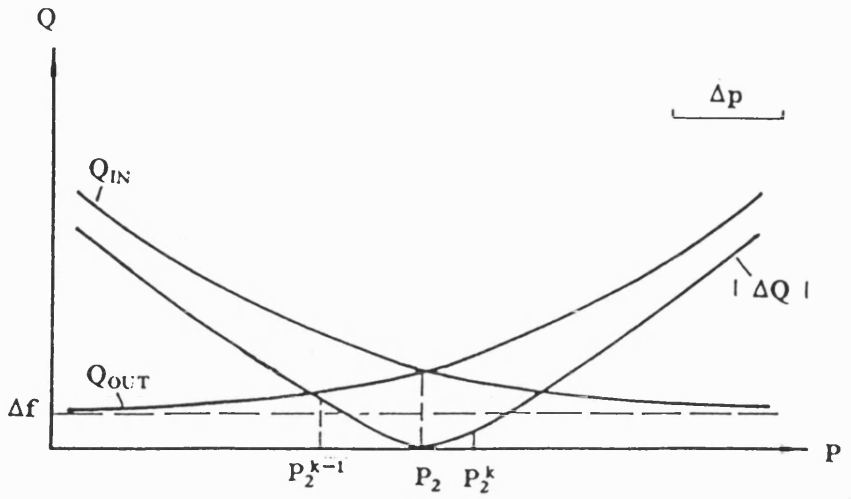
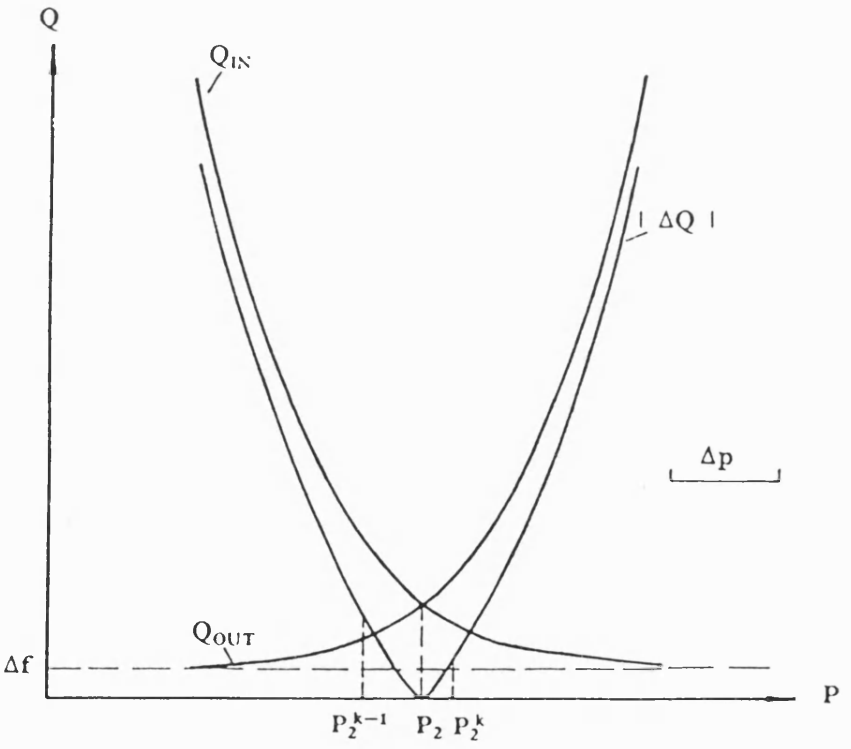


Figure 4.2 Schematic of a two stage relief valve



(a) $|\Delta Q| \leq \Delta_f$

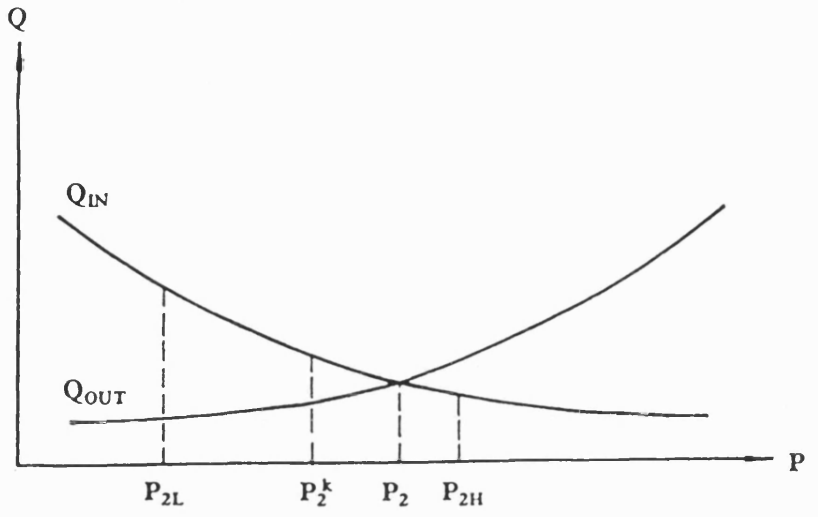
$| P_2^k - P_2^{k-1} | > \Delta p$



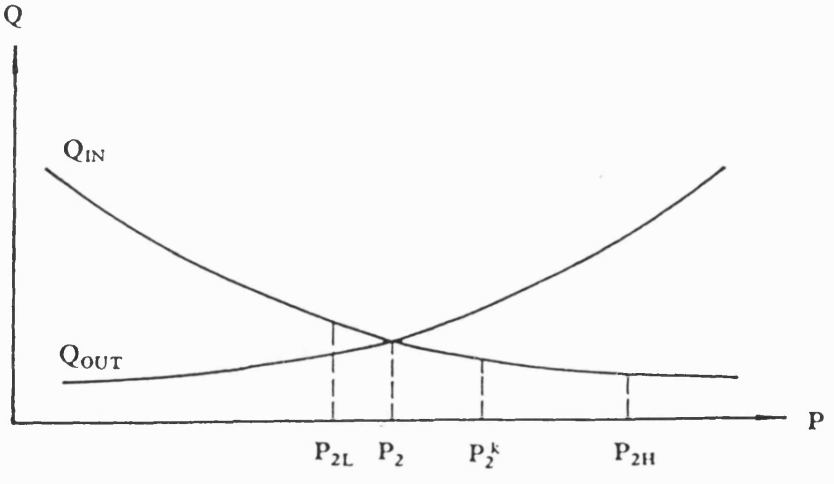
(b) $| \Delta Q | > \Delta_f$

$| P_2^{(k)} - P_2^{(k-1)} | \leq \Delta_p$

Figure 4.3 Variation of net flow with pressure in the pilot chamber of a two stage relief valve



(a) $Q_{IN} > Q_{OUT}$



(b) $Q_{IN} < Q_{OUT}$

Figure 4.4 Two situations of chosing a new halved iterative interval

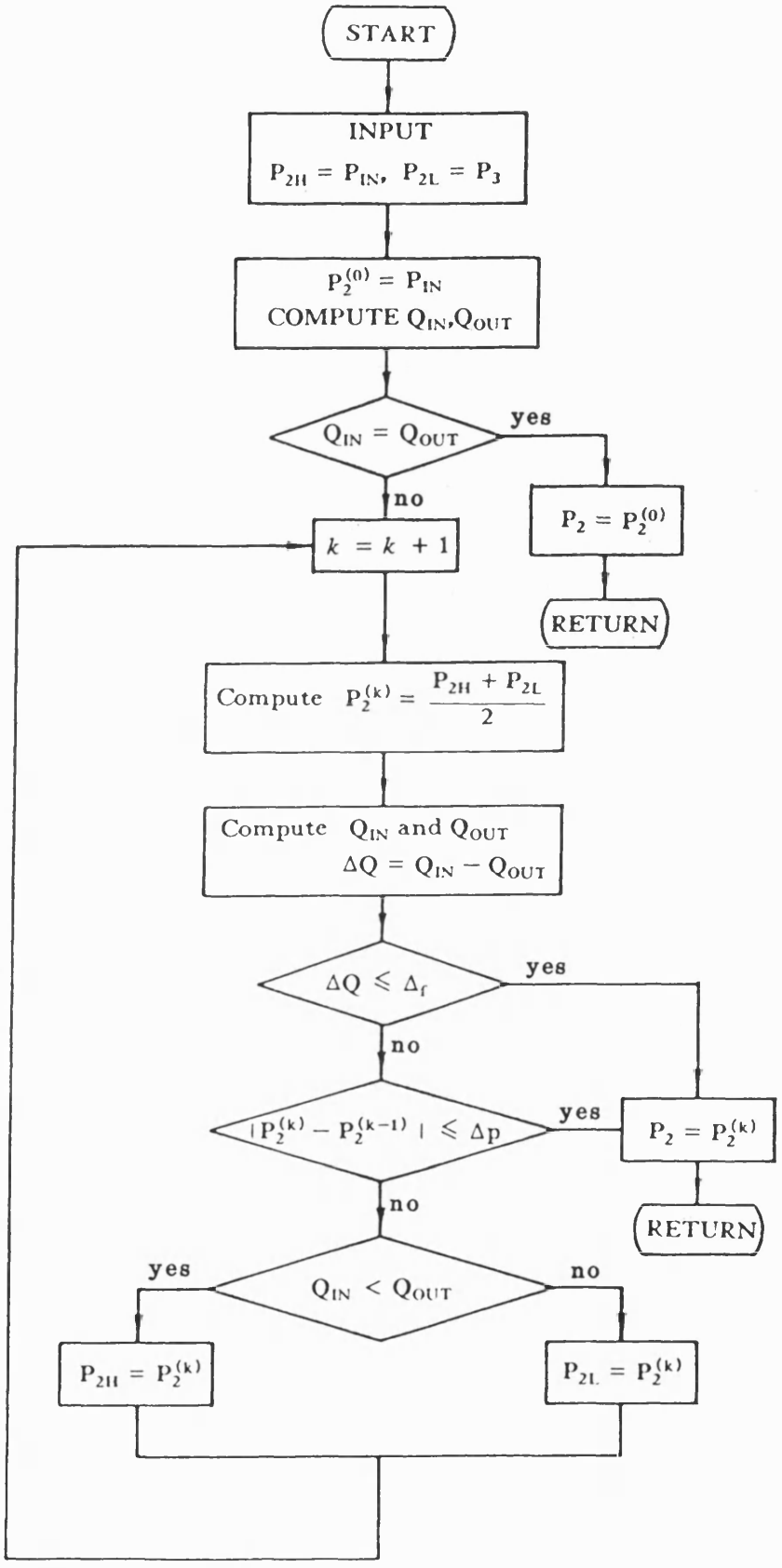


Figure 4.5 Flow chart of iterative procedure

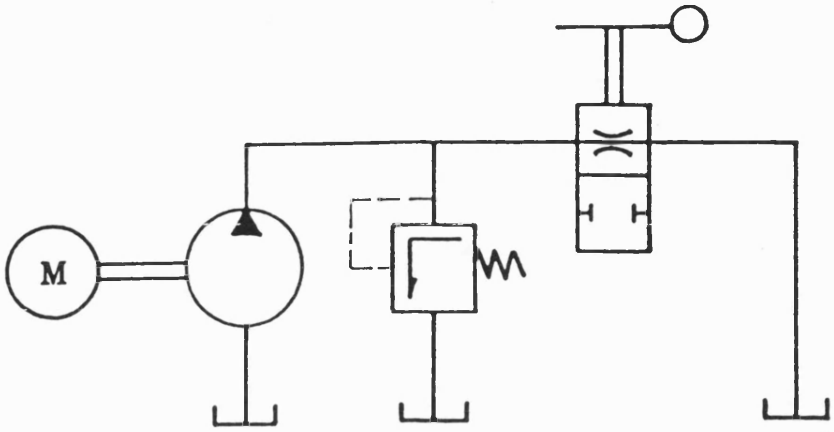


Figure 4.6 Test circuit for simulation test of model PCA1

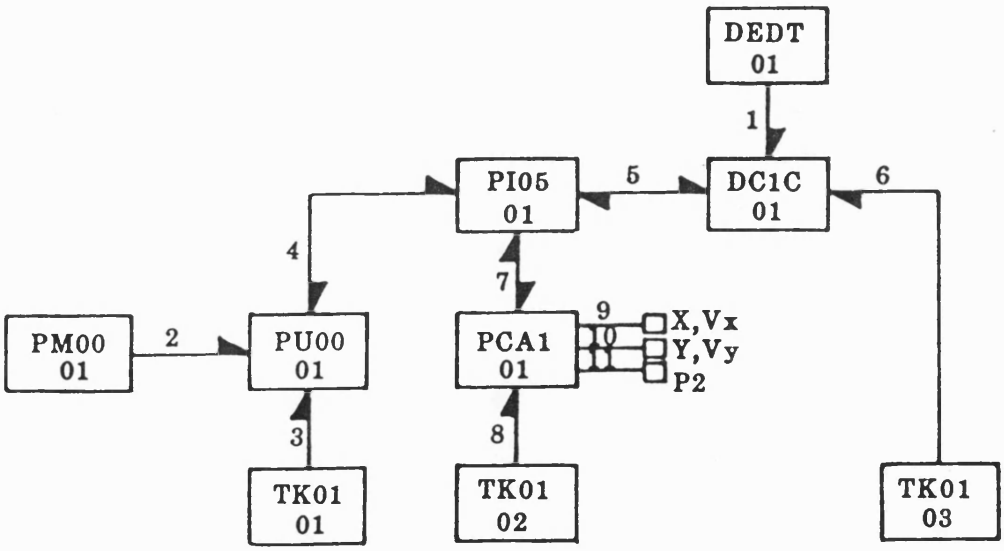


Figure 4.7 Linking diagram for test circuit of model PCA1

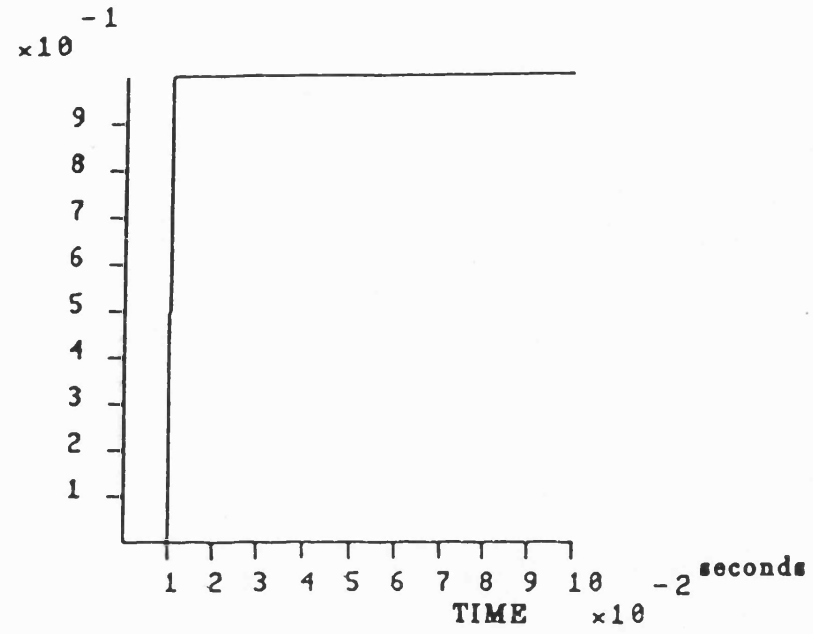


Figure 4.8 Manual position of a 2 port directional control valve

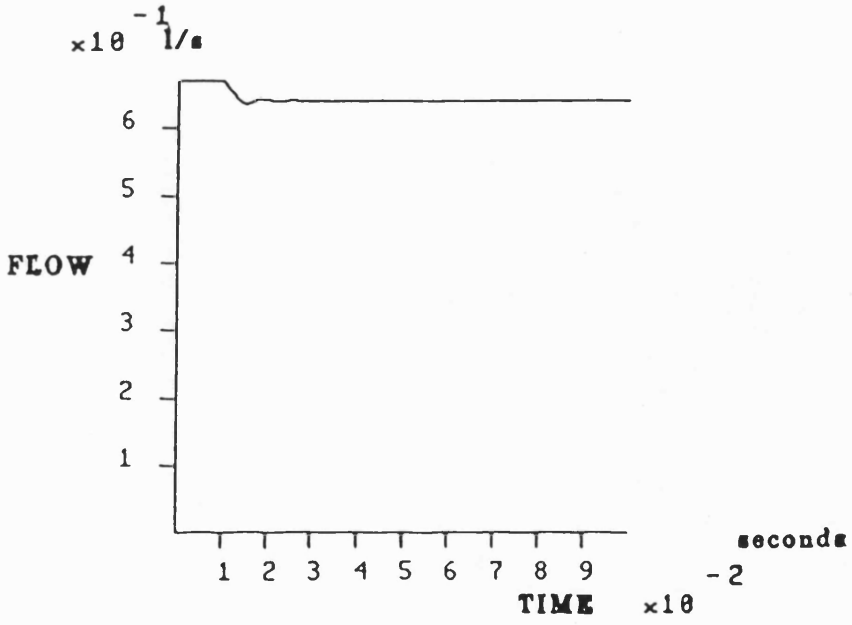
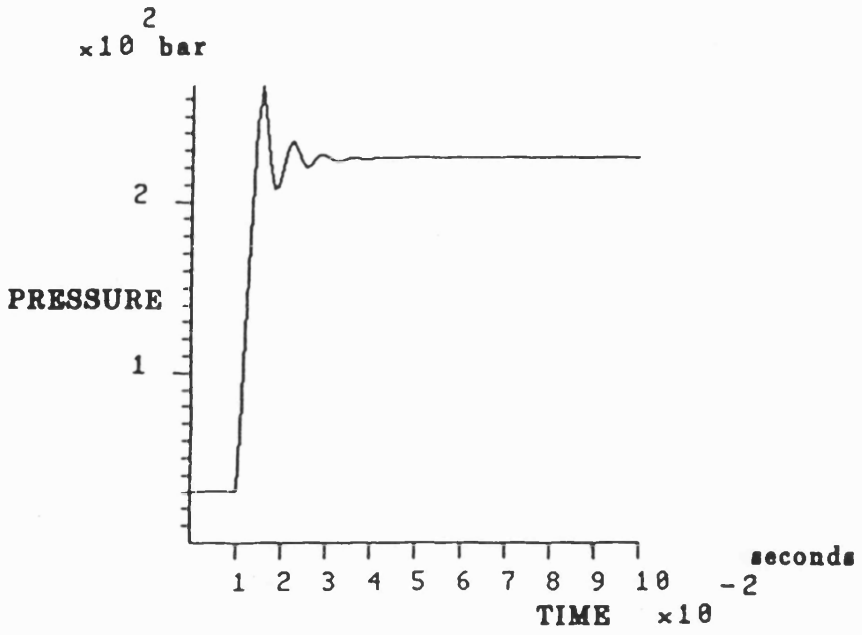
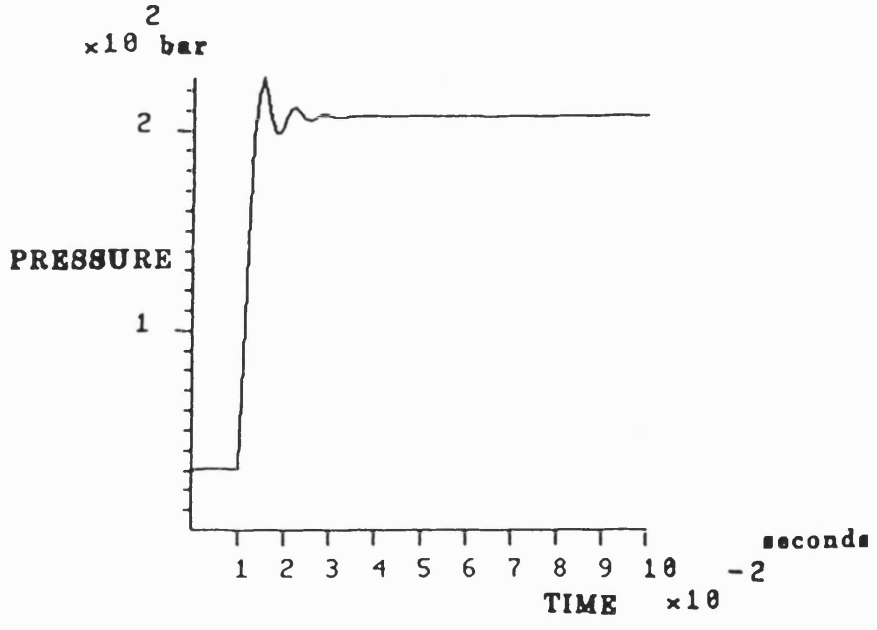


Figure 4.9 Flow rate from hydraulic pump

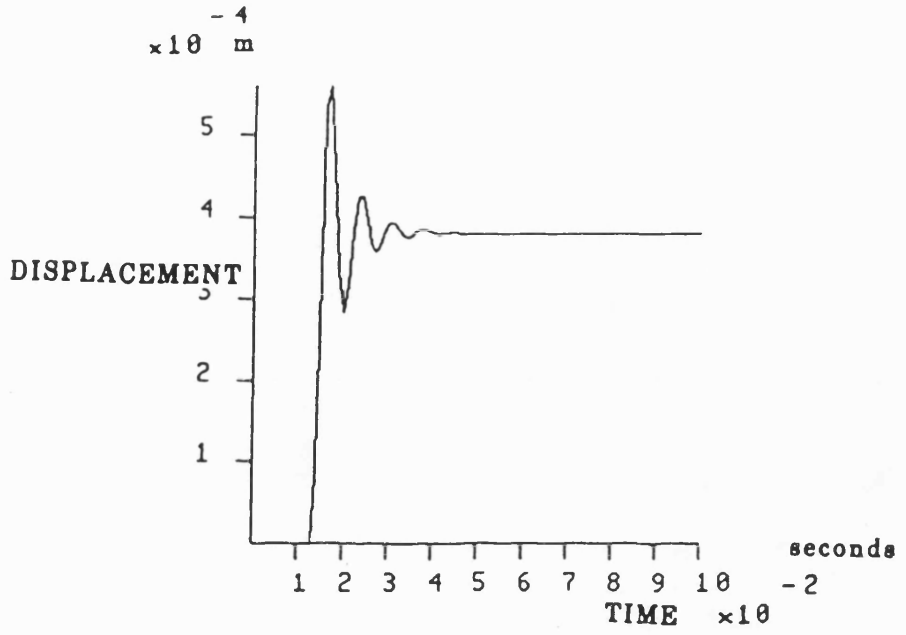


(a) Inlet pressure of valve

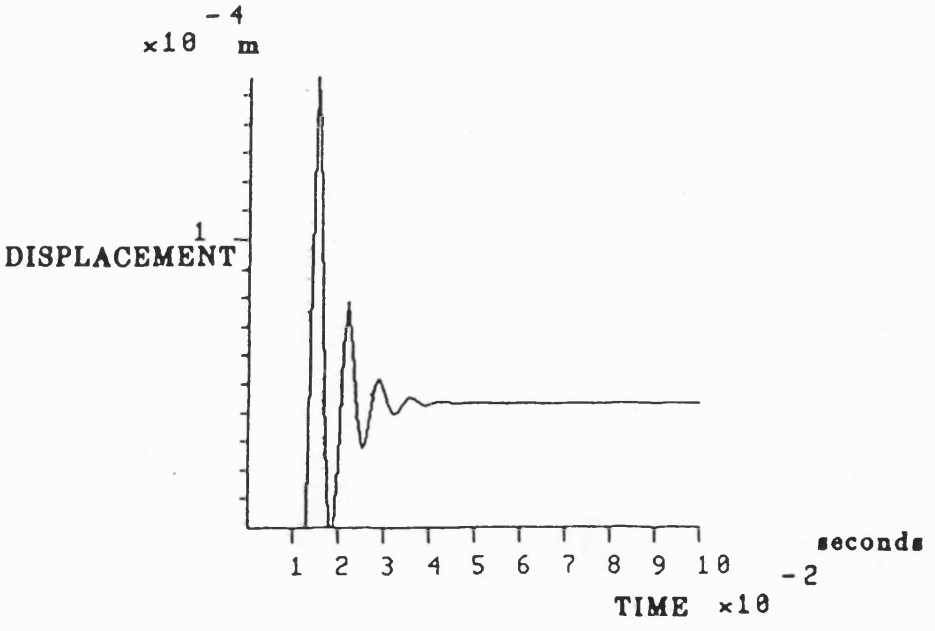


(b) Pilot pressure

Figure 4.10 Pressure responses of a two stage relief valve



(a) Main poppet displacement



(b) Pilot poppet displacement

Figure 4.11 Displacement responses of a two stage relief valve

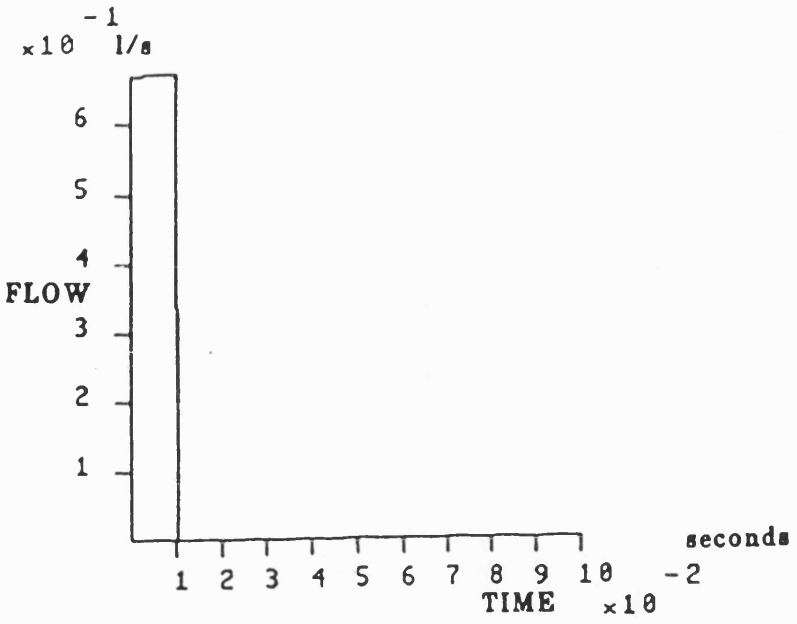


Figure 4.12 Flow rate through a DVC

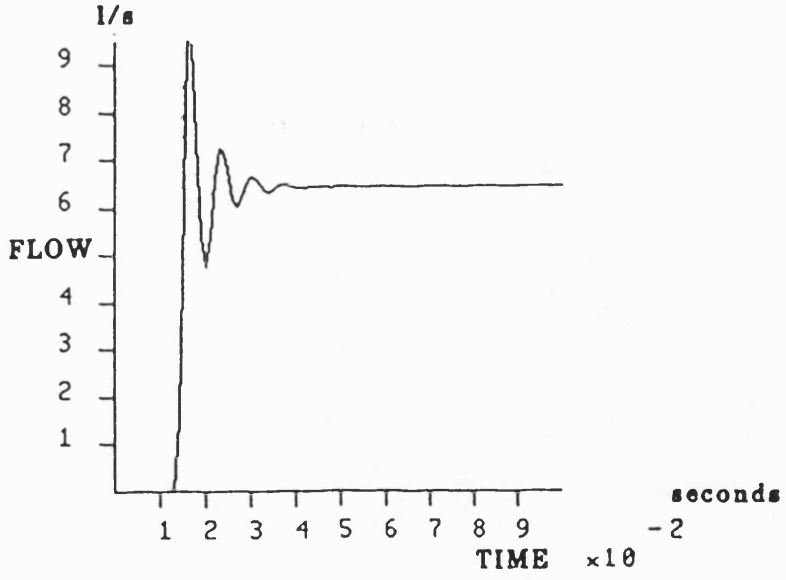
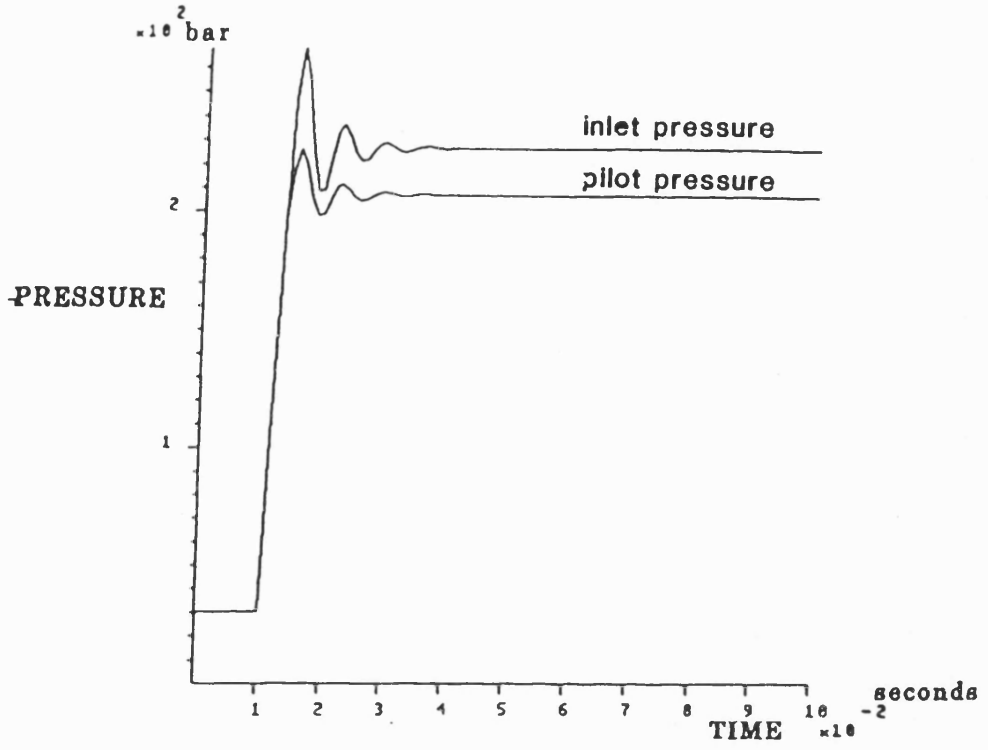
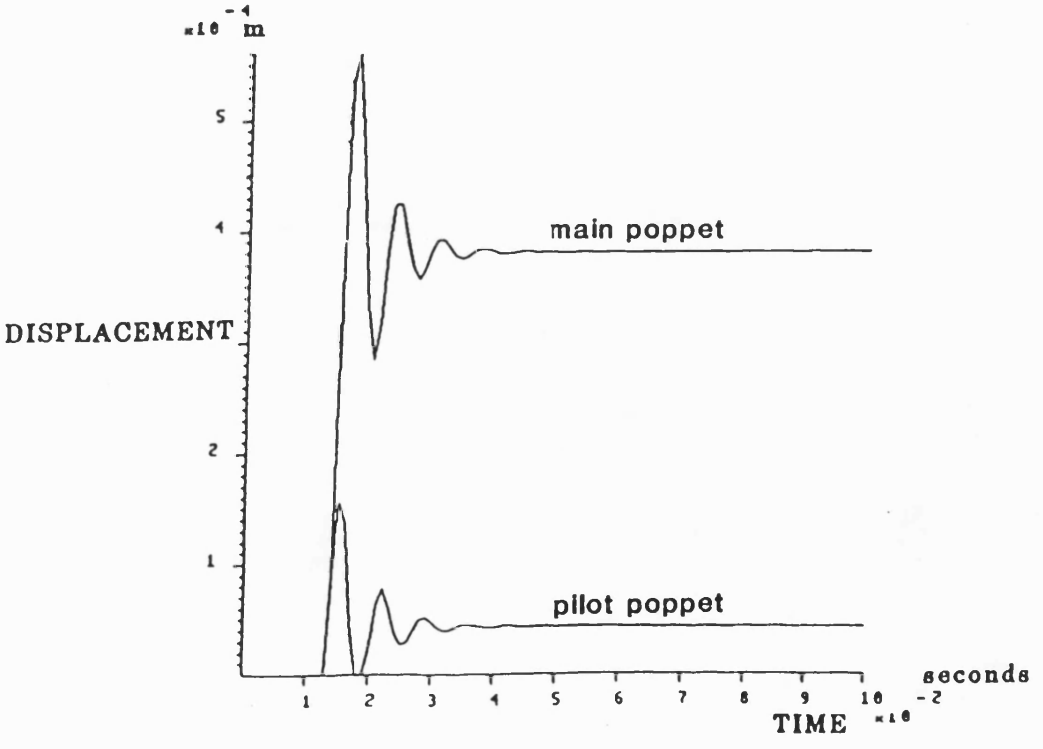


Figure 4.13 Flow rate through the two stage relief valve

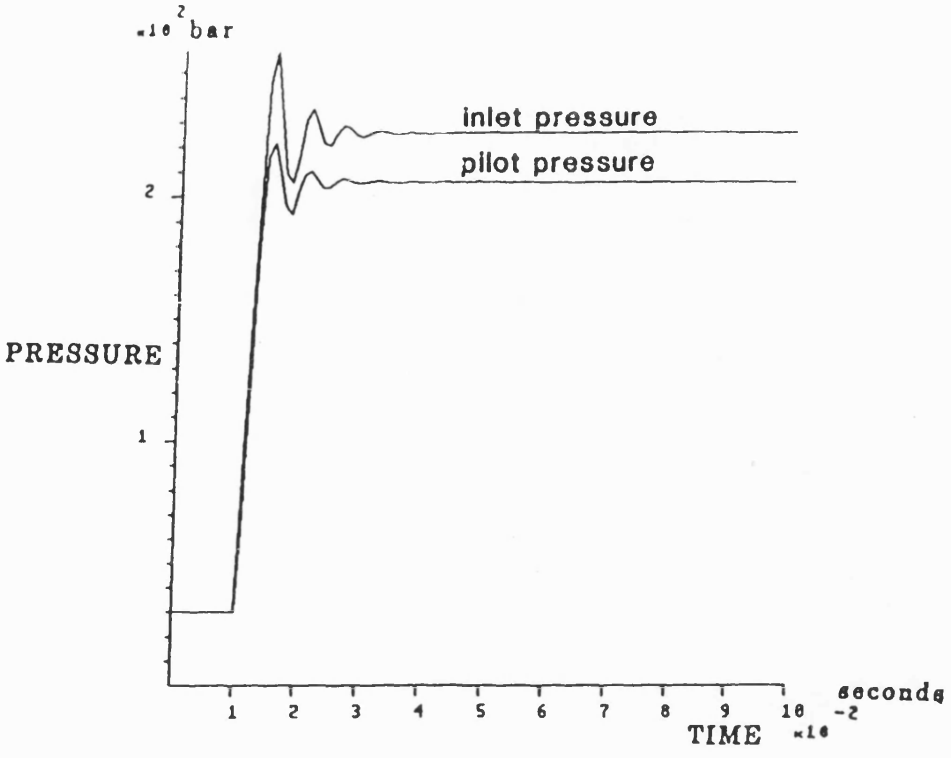


(a) Pressures

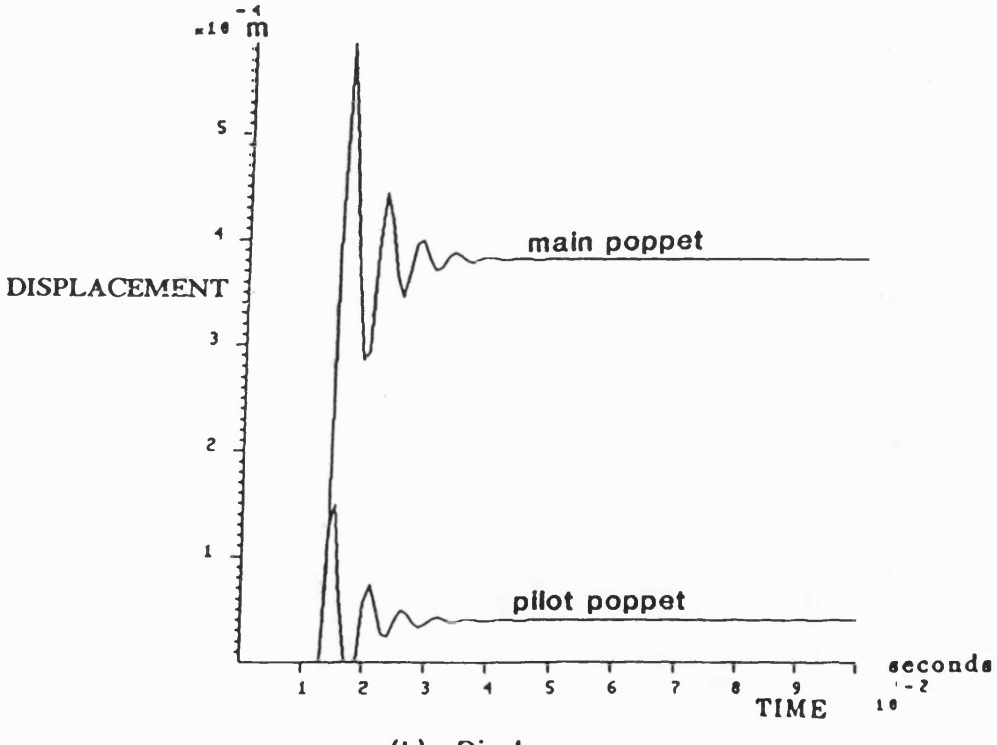


(b) Displacements

Figure 4.14 Comparisons of pressures, displacements for a two stage relief valve using model PCA1

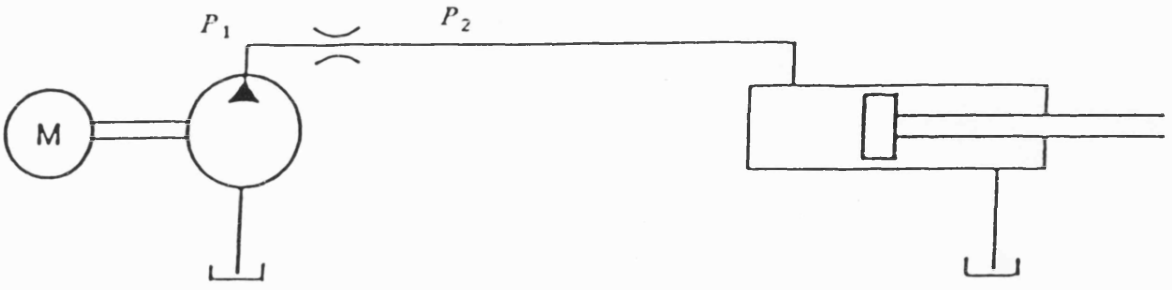


(a) Pressures



(b) Displacements

Figure 4.16 Comparisons of pressures, displacements for a two stage relief valve using model PCR1



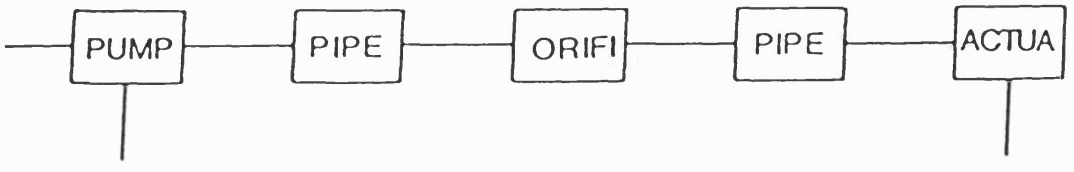


Figure 4.16 A simple pump controlled actuator circuit

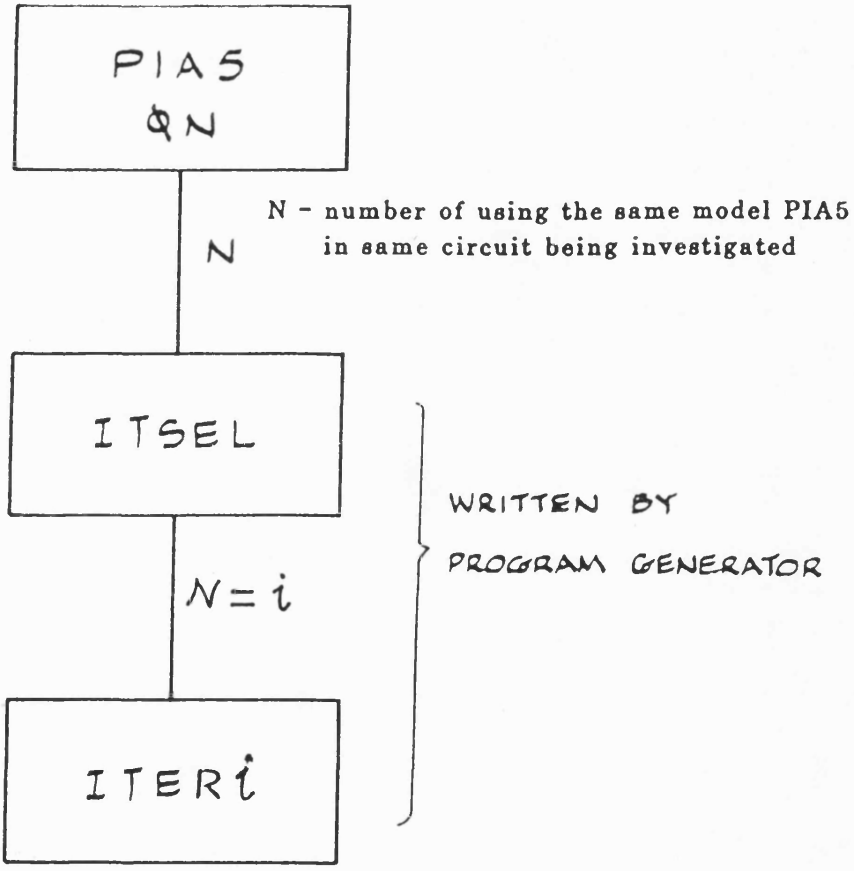


Figure 4.17 An interaction among the pipe model main section, subroutine ITSEL and the iterative procedure subroutines

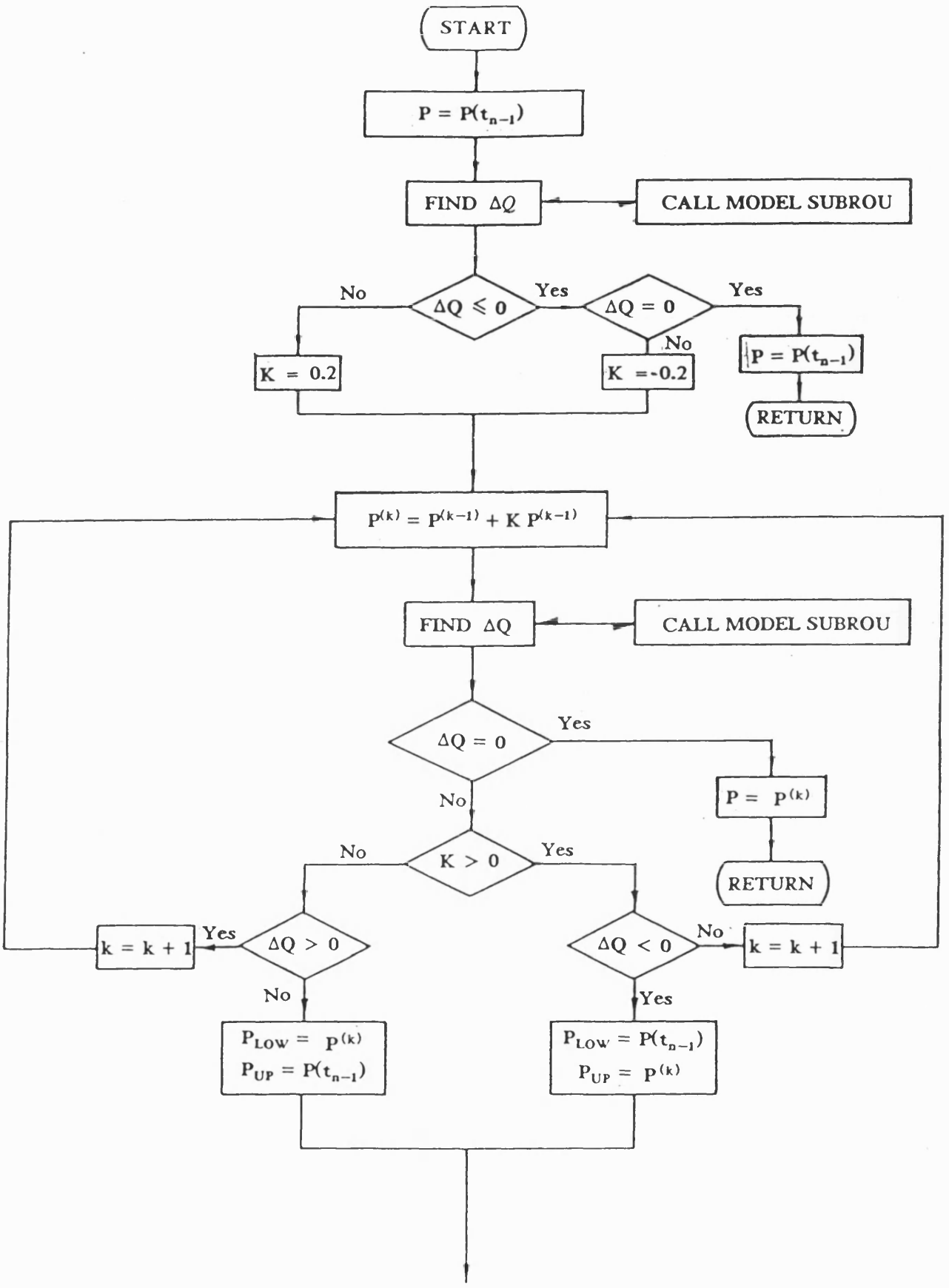


Figure 4.18 A flow chart of an automatic search procedure for finding an initial iterative interval

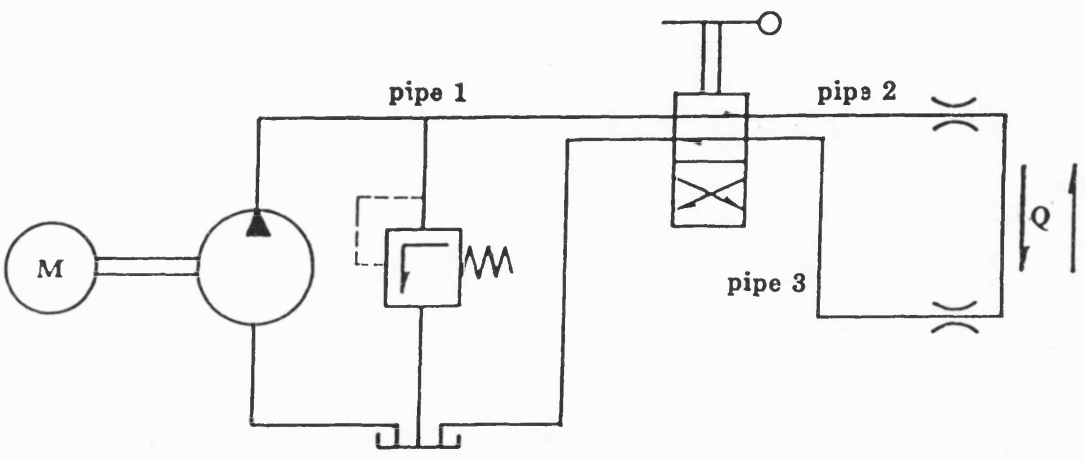


Figure 4.19 A pipe orifice system with a directional control valve

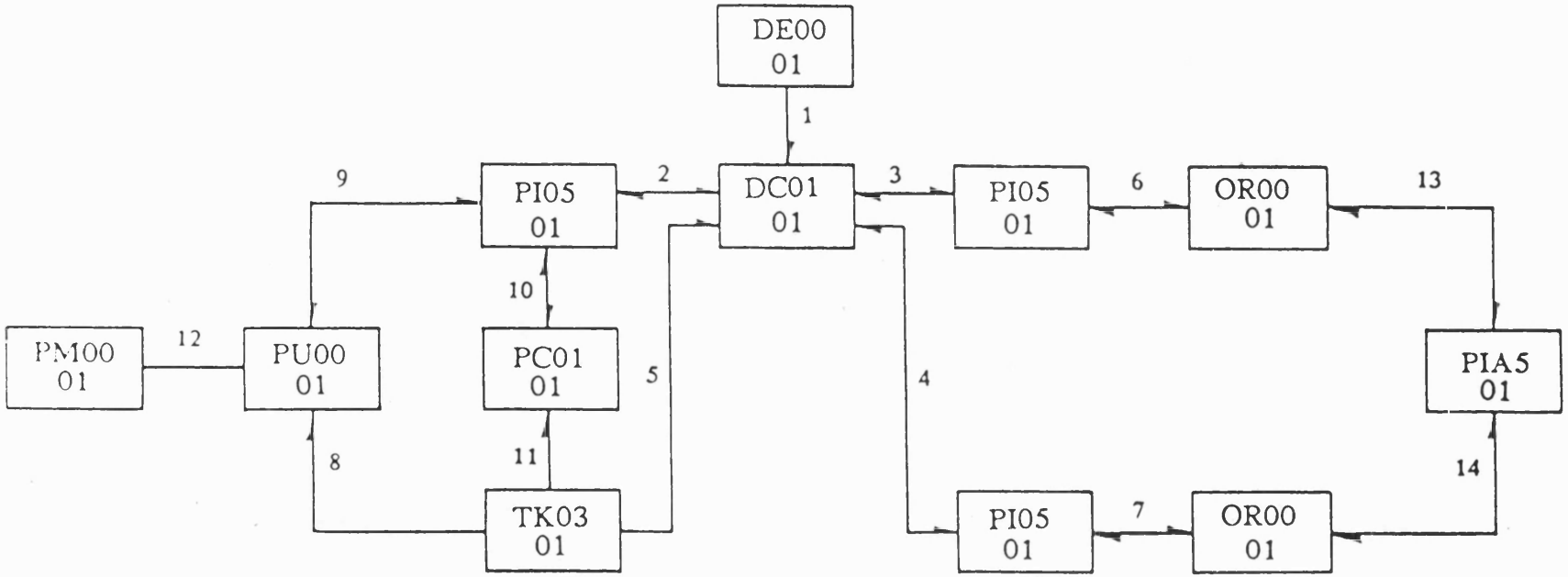


Figure 4.20 The computer link diagram of a pipe orifice system with a directional control valve

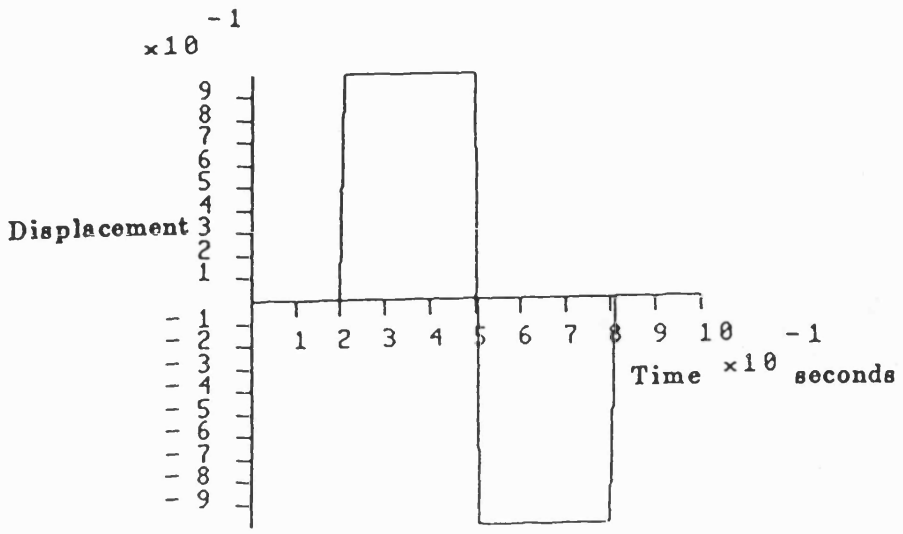


Figure 4.21 Operating characteristics of a directional control valve

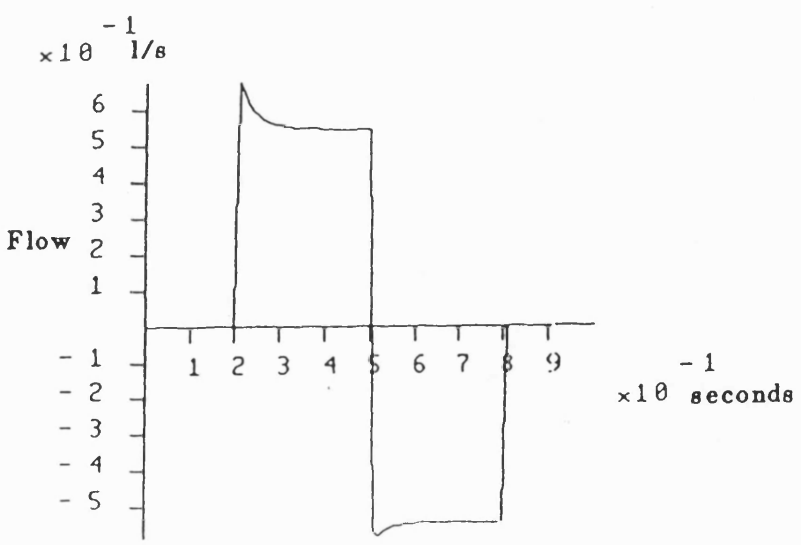


Figure 4.22 Flow characteristics of pipe between two orifices

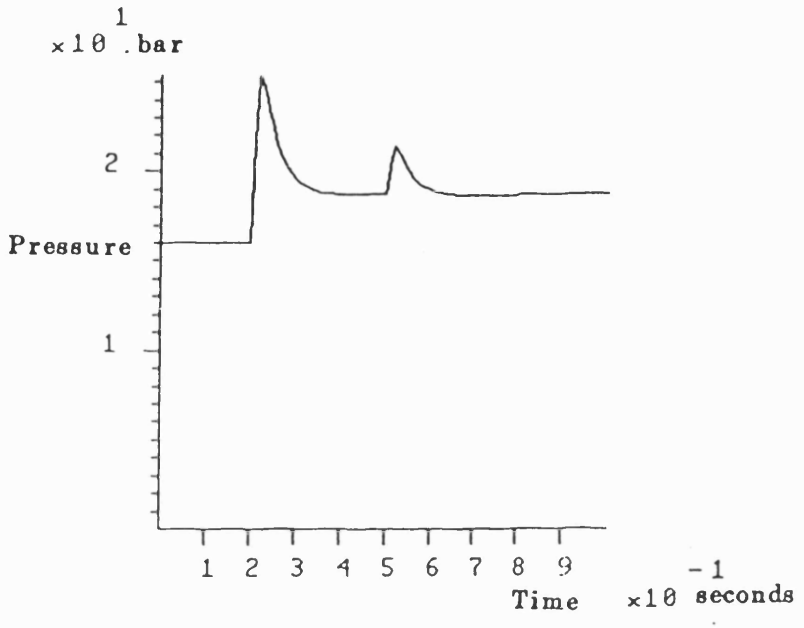


Figure 4.23 Pressure characteristics of pipe between two orifices

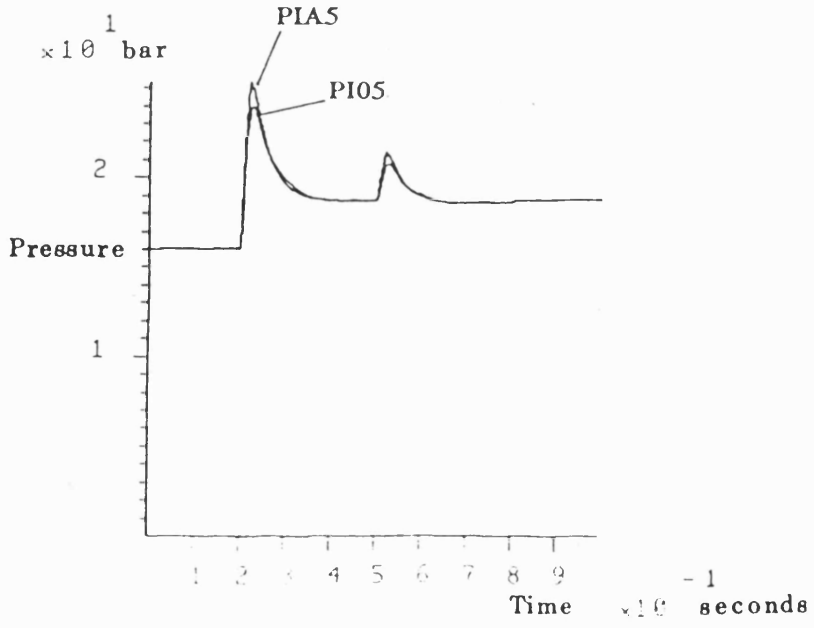


Figure 4.24 The comparison of the pressure simulation results for both systems using PIA5 and PI05

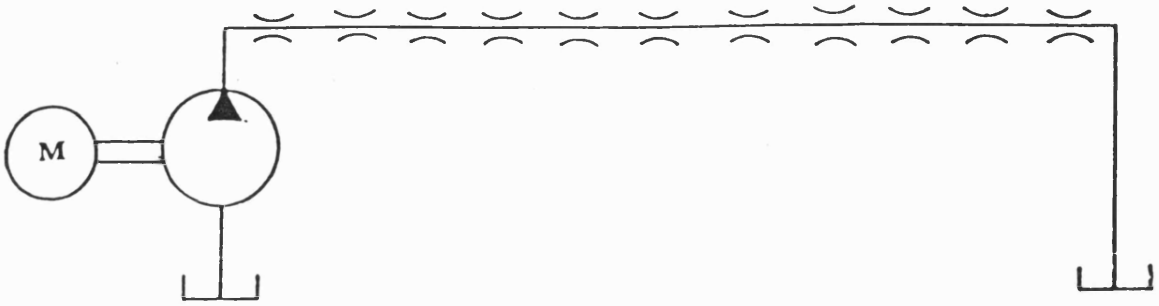


Figure 4.25 A hypothetical multi-pipe-orifice system

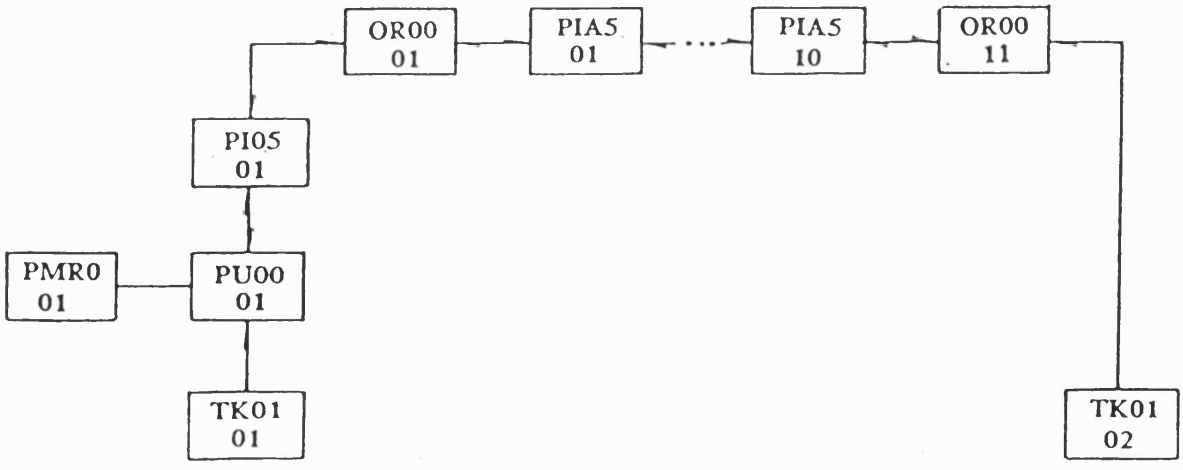


Figure 4.26 Computer linking diagram for a multi-pipe-orifice system

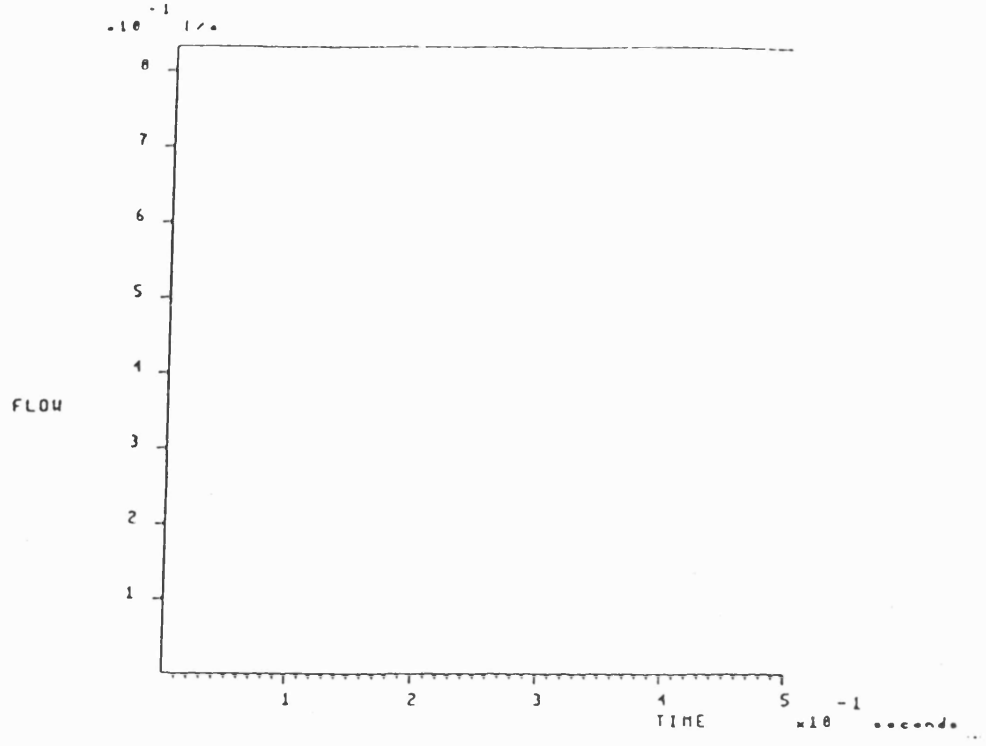


Figure 4.27 Step input flow rate of pump

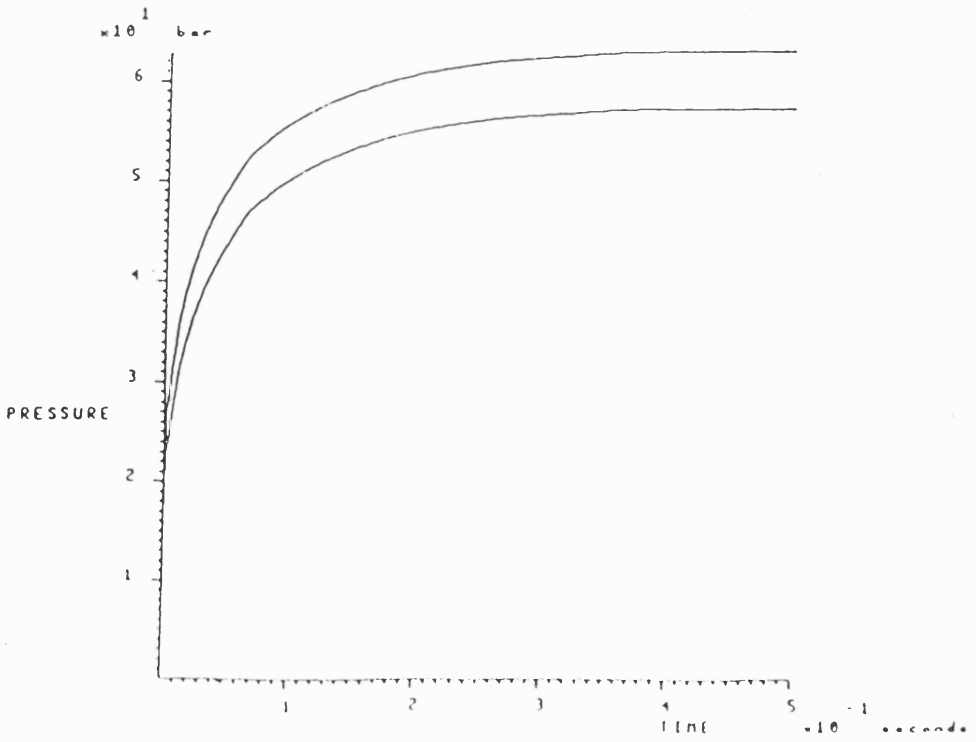


Figure 4.28 The pressure responses in pipes No.1 and No.2

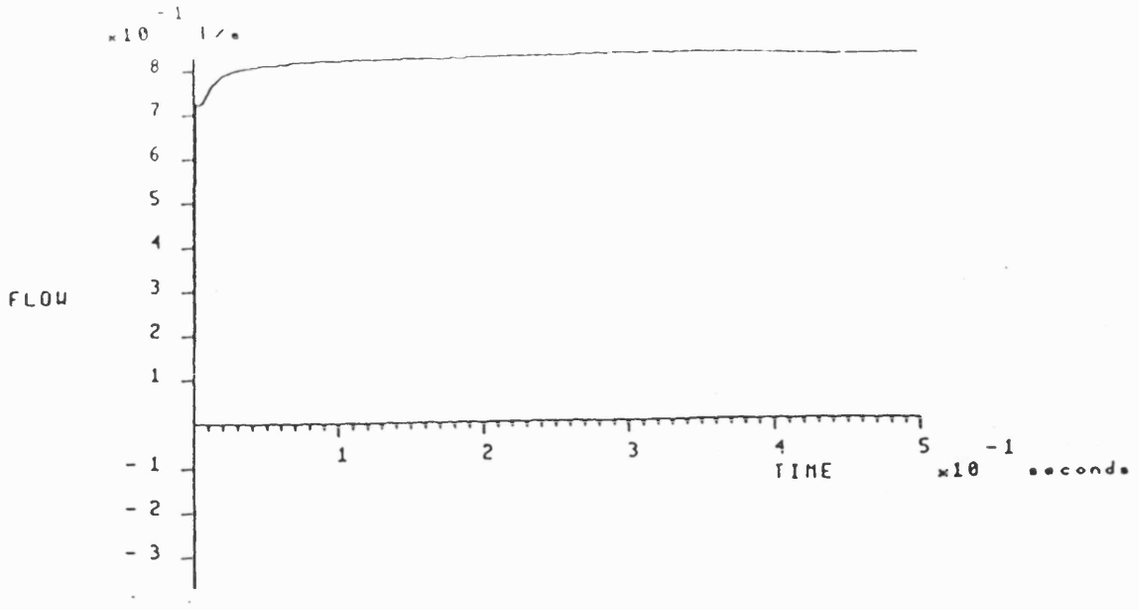


Figure 4.29 The in and out flow rates of the first orifice

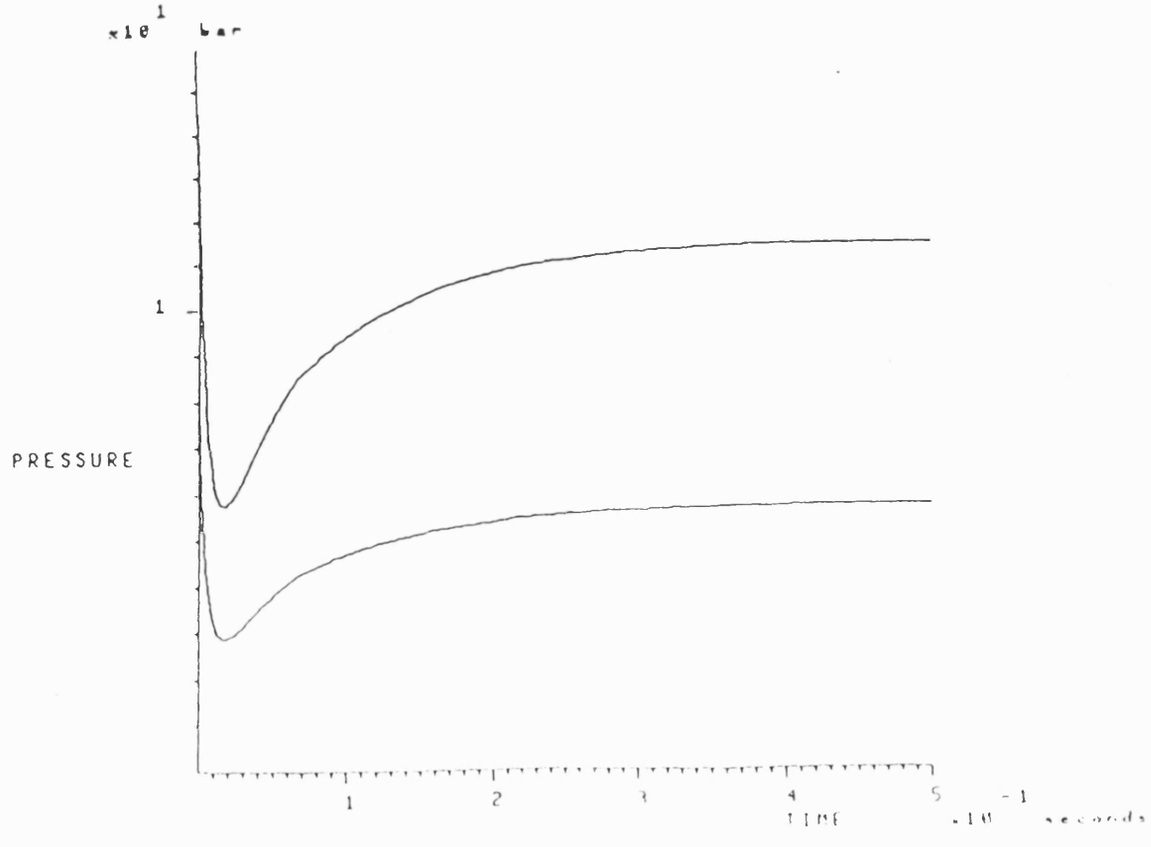


Figure 4.30 Pressure responses in pipes No.10 and No.11

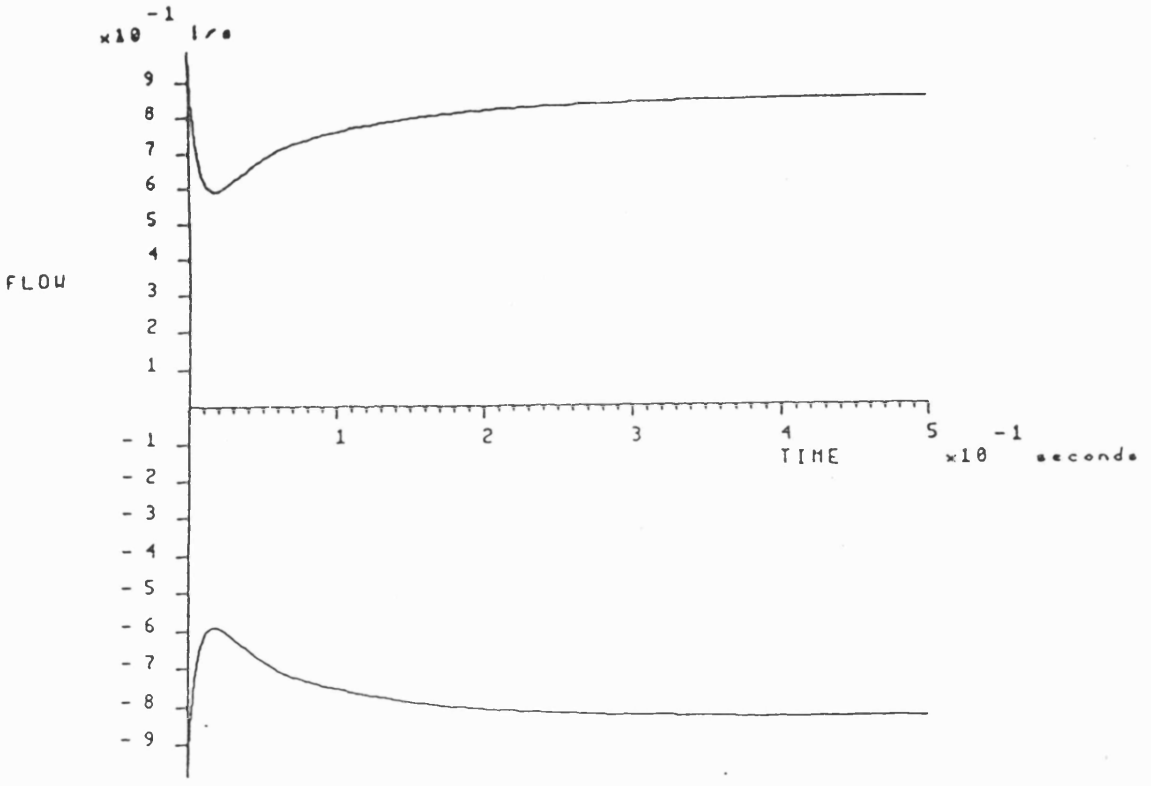


Figure 4.31 Flow characteristics of pipes No.10 and No.11

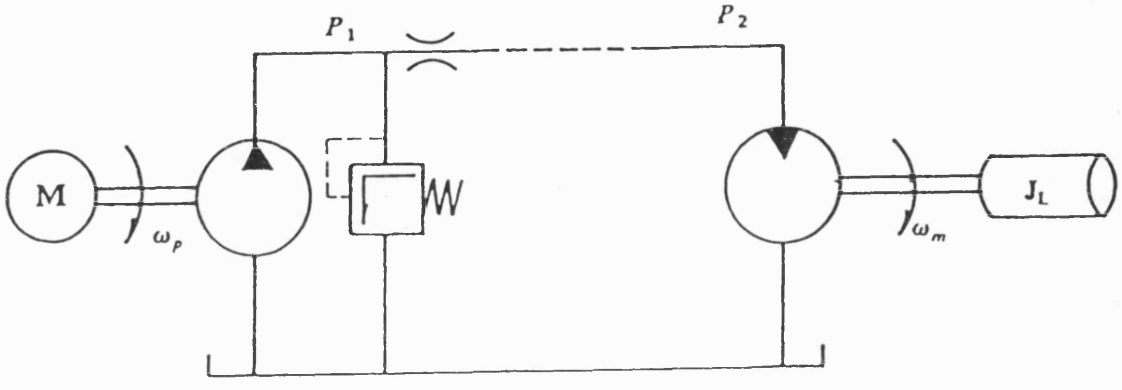


Figure 4.32 A simple hydrostatic transmission with an orifice

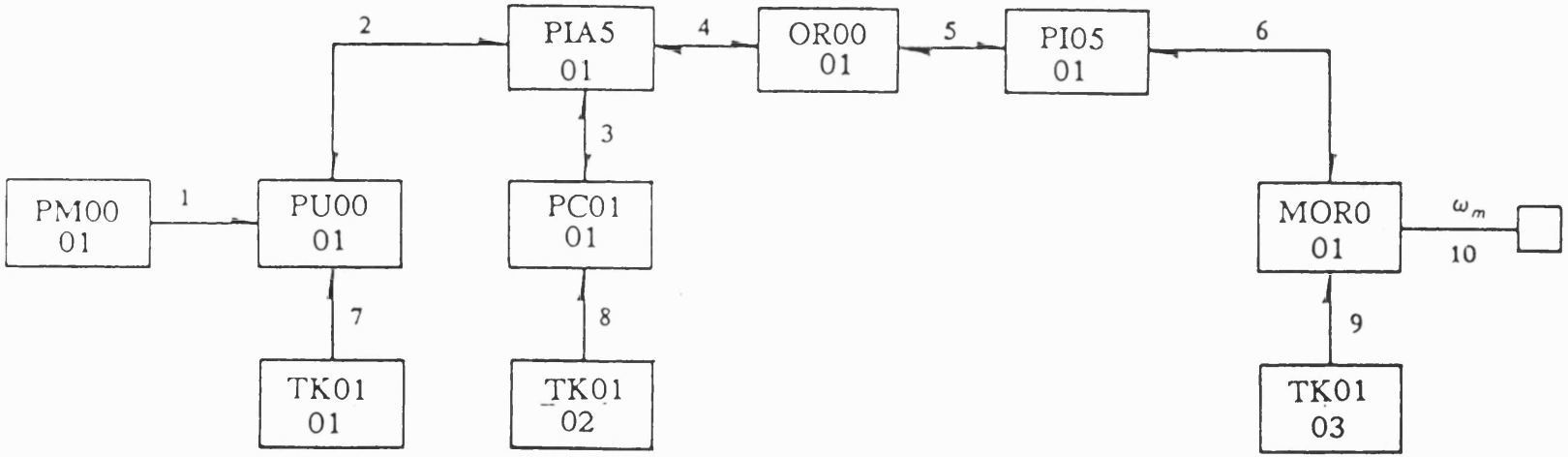


Figure 4.33 Computer linking block diagram of a simple hydrostatic transmission

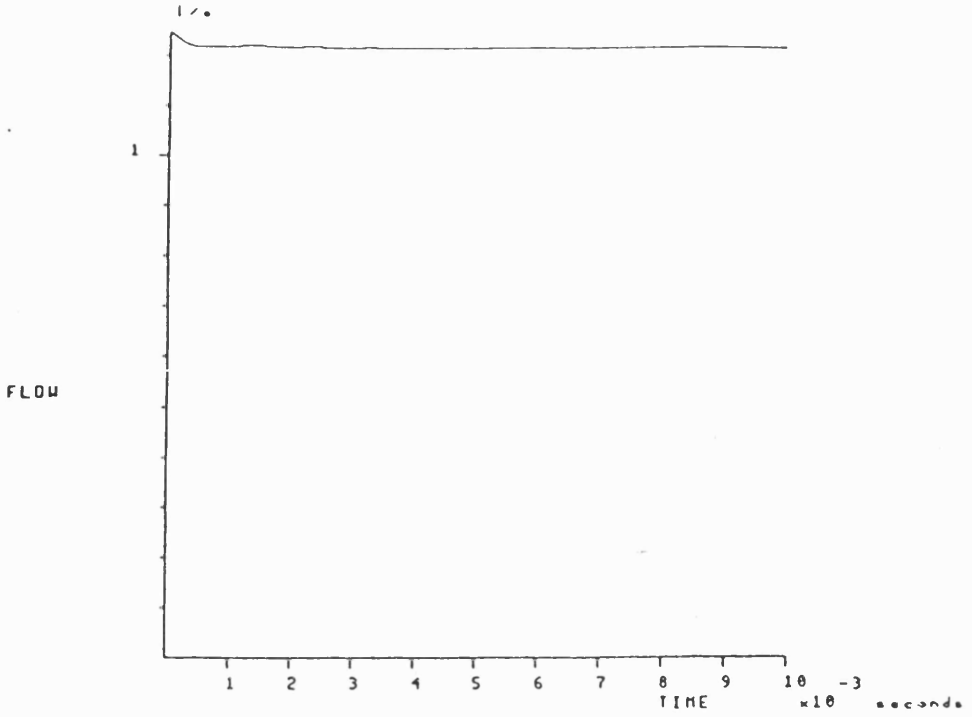


Figure 4.34 Flow rate of the pump

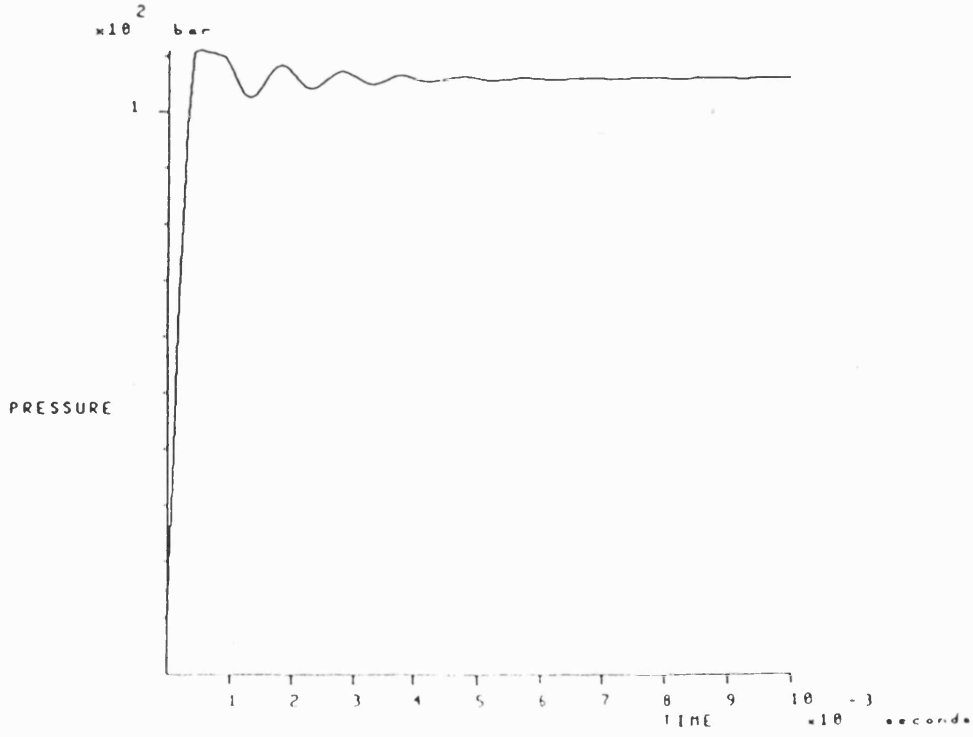


Figure 4.35 Pressure characteristics of pipe No.1

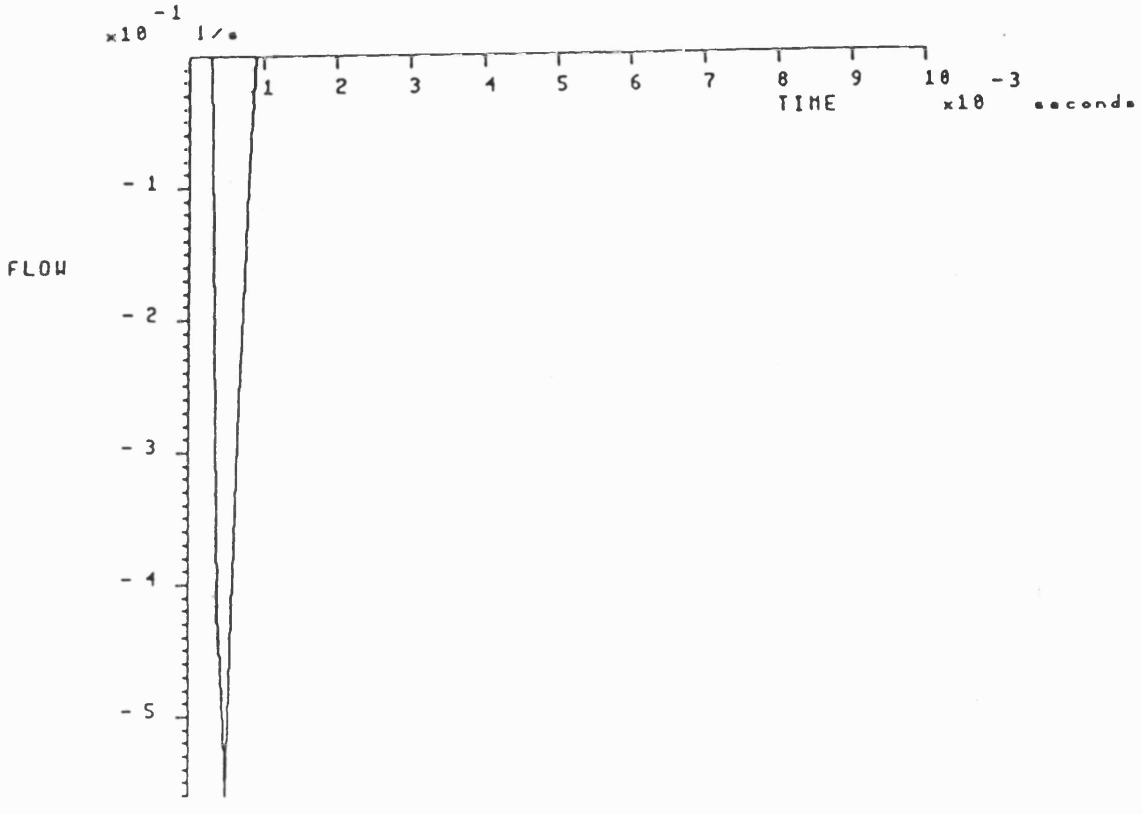


Figure 4 36 Flow rate through the relief valve

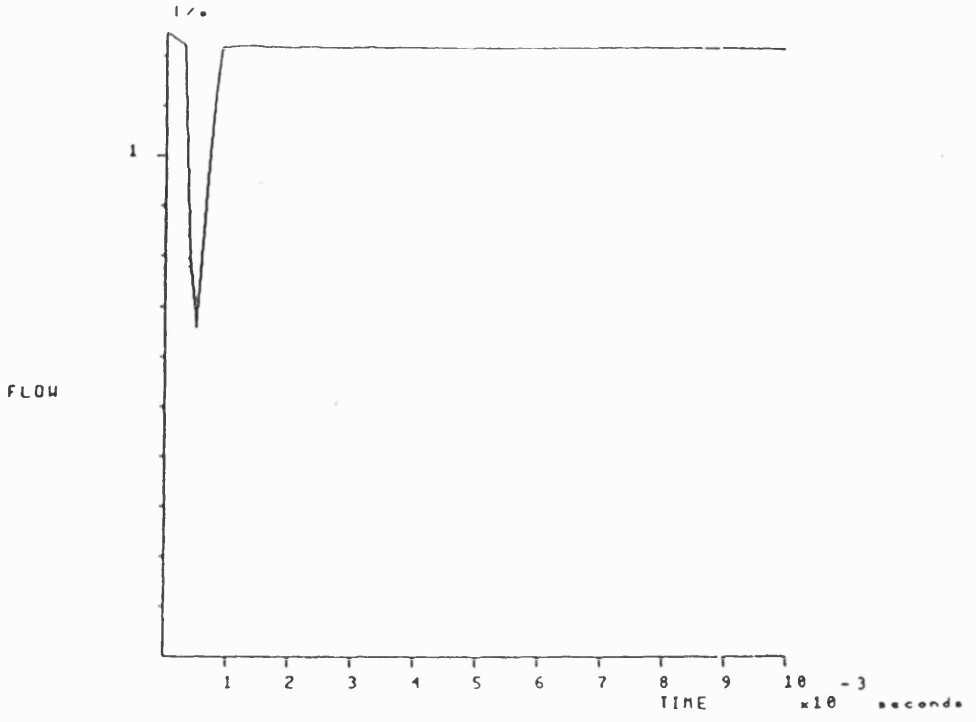


Figure 4.37 Flow rate to motor

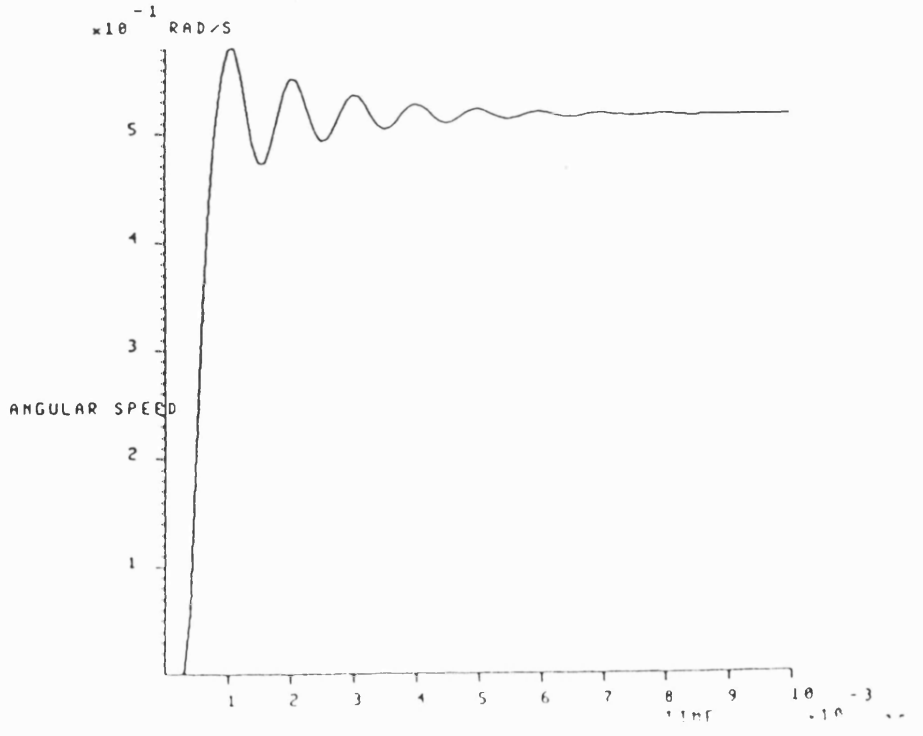


Figure 4.38 Motor angular speed

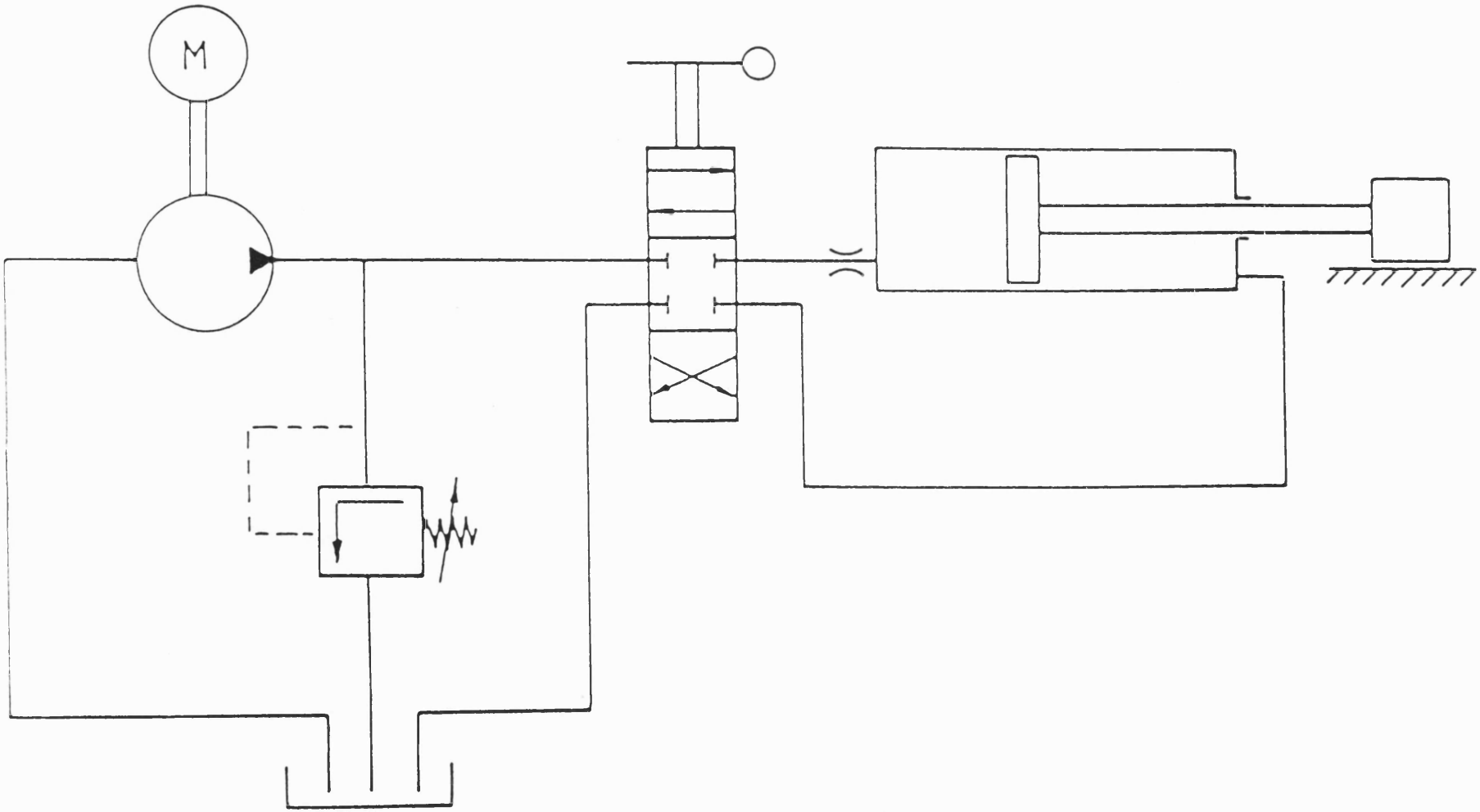


Figure 4.39 A linear actuator system with a control orifice

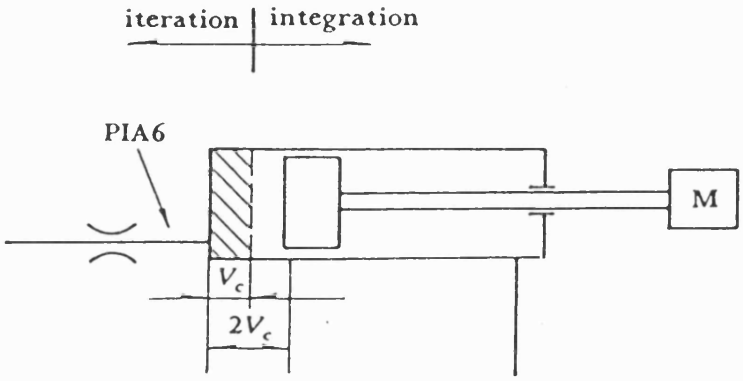


Figure 4.40 Schematic of operation regions of iteration and integration

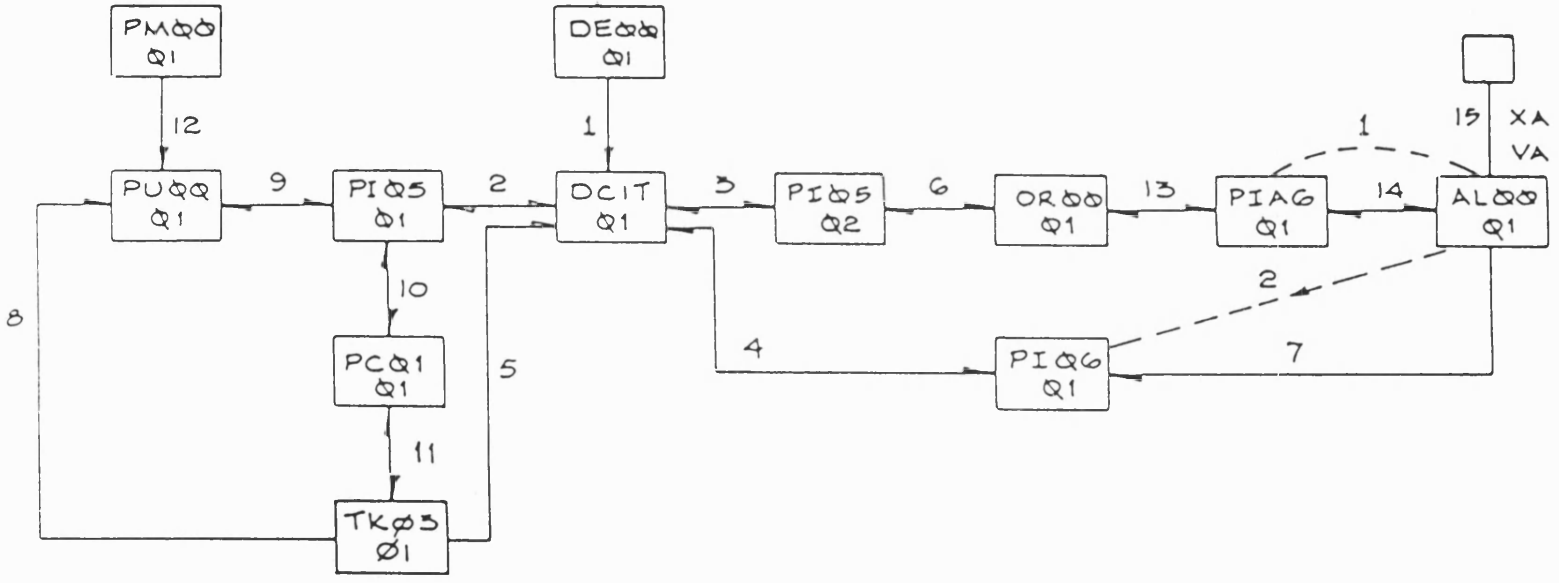


Figure 4.41 Computer model linking diagram of a linear actuator system with a control orifice

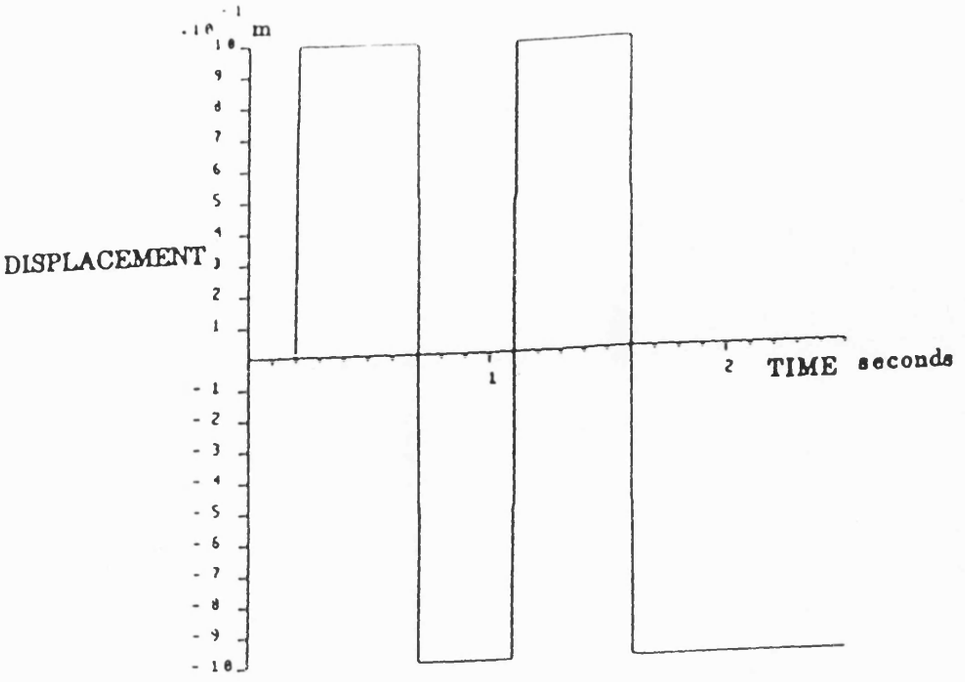


Figure 4.42 DCV manual position for model (PIA6) simulation test

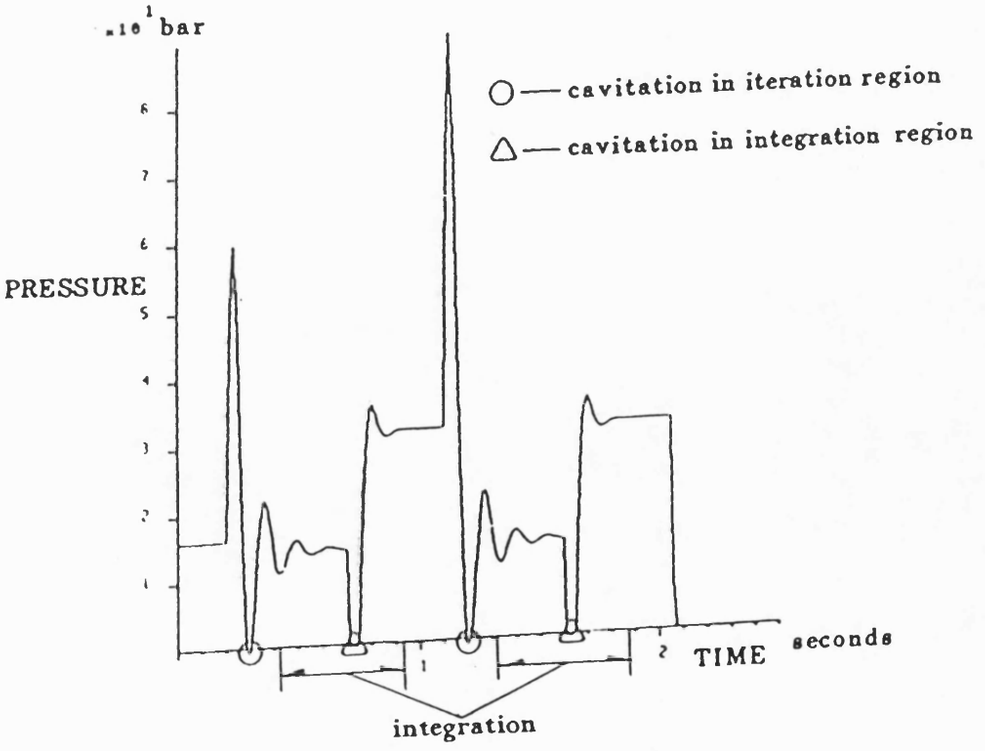


Figure 4.43 Pressrue characteristics of actuator piston chamber

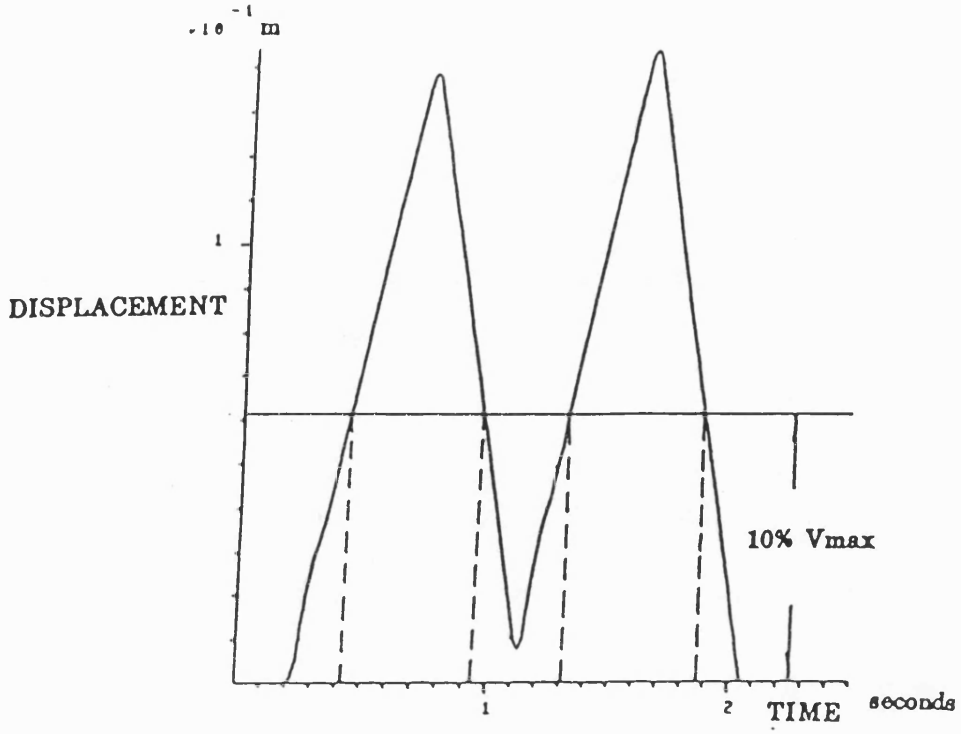


Figure 4.44 Actuator displacement obtained in model (PIA6) simulation test

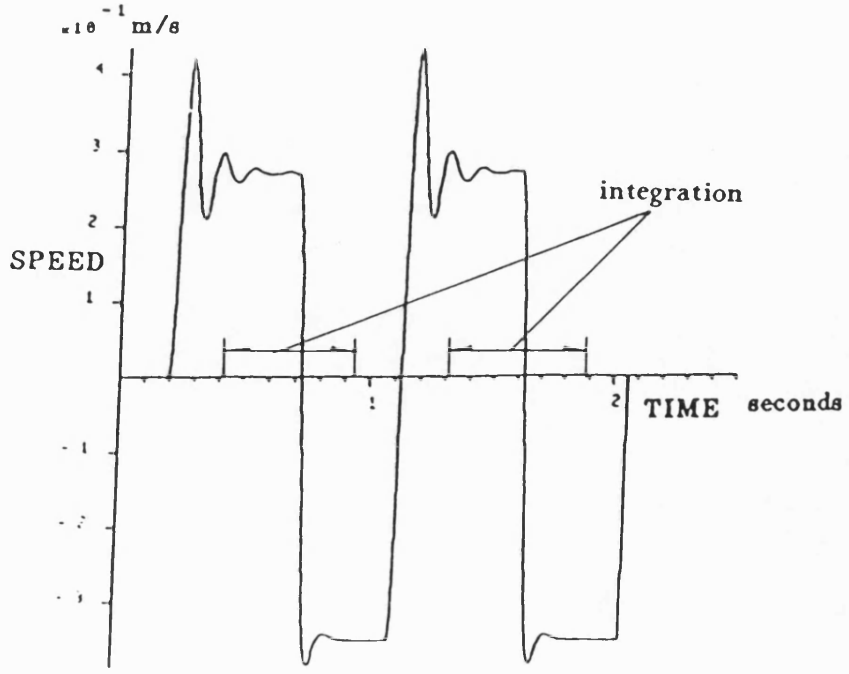


Figure 4.45 Actuator velocity obtained in model (PIA6) simulation test

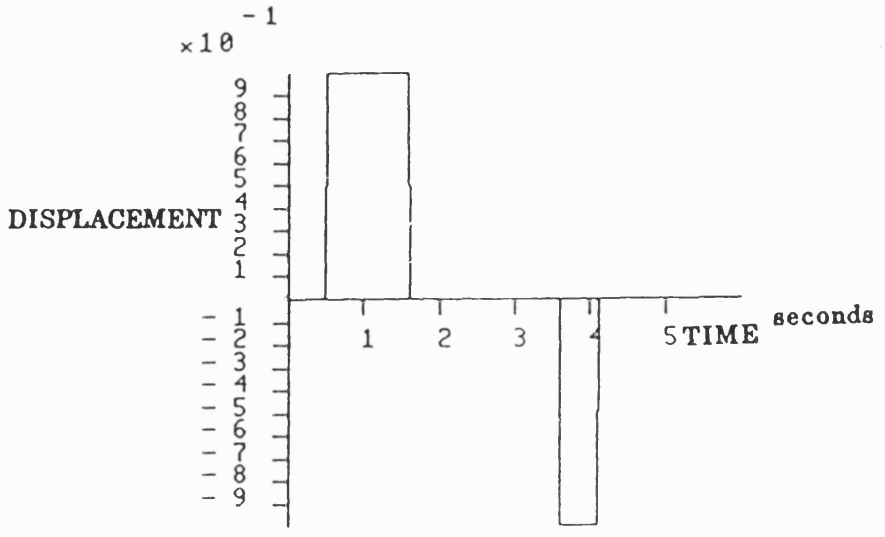


Figure 4.46 DCV manual position in simulation of linear actuator circuit

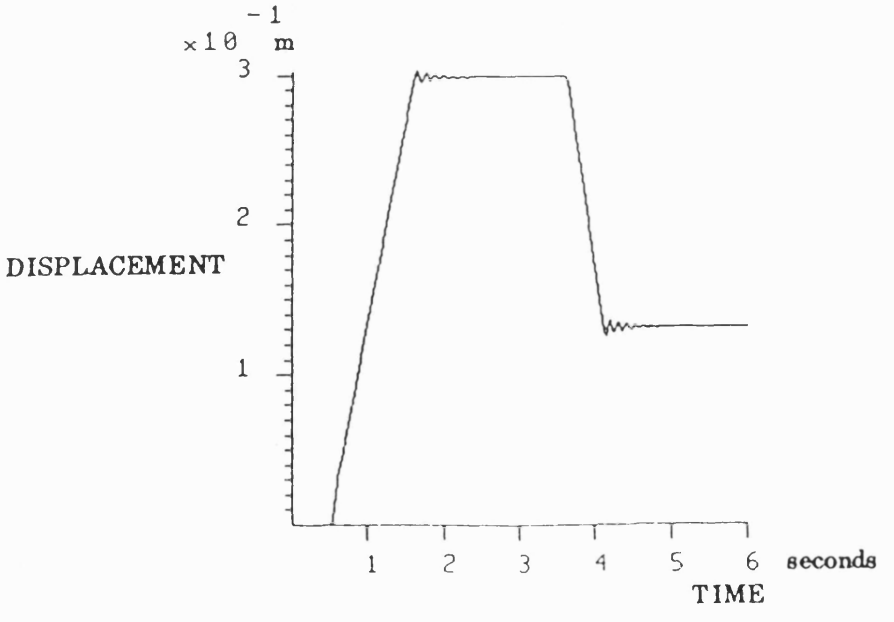
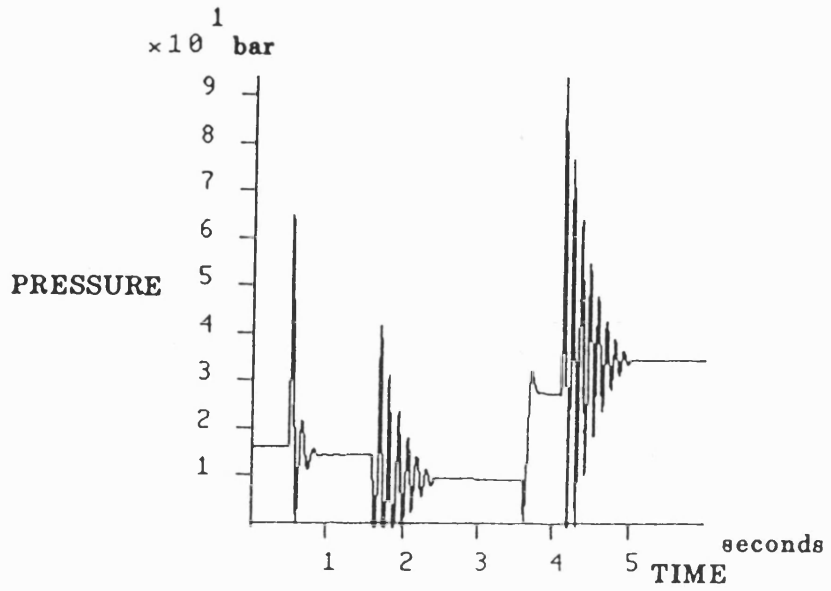


Figure 4.47 Actuator displacement



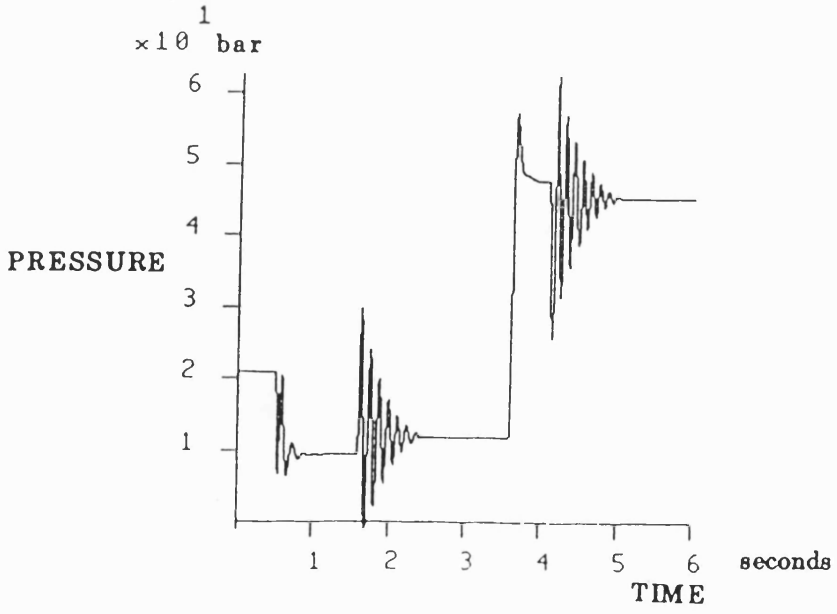


Figure 4.48 Presssure responses in both chambers of actuator

CHAPTER 5

ITERATIVE TECHNIQUE FOR AVOIDING THE STIFFNESS PROBLEMS IN THE SIMULATION OF SHAFT COUPLED POWER TRANSMISSION SYSTEMS

INTRODUCTION

5.1 Digital simulation for the transmission of power from prime movers to a variety of mechanical loads is an attractive and efficient tool for transmission engineers. Since the transmission of power in a hydraulic transmission system such as that shown in figure 5.1 is carried out through mechanical shafts, the modelling of a mechanical shaft associated with power transmission will affect the simulation of whole behaviour of the transmission system.

5.2 **Simulation difficulty of normal shaft model.** The normal mathematical model describing the shaft dynamics could cause mathematical stiffness problems in the simulation of systems due to the very high rigidity modulus of shaft. For instance, a circular shaft can be described by a differential equation given as follows:

$$\frac{dT_s}{dt} = \frac{G\ I}{l}(\omega_D - \omega_L) \qquad \ldots\ldots (5.1)$$

where T_s - torque acting on the shaft in Nm

G - the modulus of rigidity in GN/m^2

I - the polar moment of inertia in $\frac{\pi D^4}{32}$

l - the length of shaft in m

ω_D - shaft angular speed of driving side in 1/s

ω_L - shaft angular speed of loading side in 1/s

Since the product $\frac{G\ I}{l}$ is usually large, a long computational time would be expected.

5.3 It is often the case that a shaft model is ignored and the load model is directly linked with a prime mover model. For example, in the simulation of hydraulic transmission circuits using HASP, the pump model which is the load of a prime mover can be directly linked to a prime mover model without a shaft model between them and the hydraulic motor model can be linked to a load model. However, this can only be done by lumping the inertia of the pump and prime mover or hydraulic motor and load together[49]. This is a limitation in the case where the effective inertia can vary at different times in a simulation. For instance, in a simple hydraulic winch system shown in figure 5.2, the continuously variable inertia is involved due to the change of cable coil size in the cable drum during the raising or lowering action of the winch. Therefore, we need develop individual shaft models which can link with adjacent models and involve the effects of effective variable inertia and at the same time avoid the stiffness difficulty mentioned above.

BASIC STRUCTURE OF INDIVIDUAL SHAFT MODEL INCORPORATING AN ITERATIVE PROCEDURE

5.4 Consider a motor shaft-load circuit shown in figure 5.3. The rigidity modulus in the shaft which is linked with the motor and load is very high, hence an iterative procedure can be used to instantaneously compute the torque of the shaft and thus avoid integration problems.

5.5 Since we only consider the instantaneous torque value of shaft, the angular speeds, accelerations and torques acting on both shaft input side and output side

must be identical, i.e.

$$\omega_D = \omega_L$$

$$\dot{\omega}_D = \dot{\omega}_L$$

$$Ts_1 = Ts_2$$

5.6 The technique used in individual shaft model is to obtain an instantaneous torque value for T_s and the shaft angular acceleration $\dot{\omega}$ by iteration, and then obtain the shaft angular speed by integration. The basic structure of the shaft model should include:

a) iteration for shaft torque

b) iteration for angular acceleration

c) integration for angular speed

This model requires torque and inertia values from adjacent models and supplies the angular speed to them. The angular speed is a state variable, and hence the shaft model is a memory model and there is no linking difficulty between this and any adjacent model.

PHILOSOPHY OF OBTAINING SHAFT TORQUE, ANGULAR ACCELERATION AND ANGULAR SPEED VALUES

5.7 Consider a motor-shaft-load subcircuit of a hydrostatic transmission as shown in figure 5.3. The shaft is assumed to be rigid. The motor driving torque T_D, load torque T_L and the relevant motor and load inertias are known and supplied from adjacent models. The initial acceleration on the shaft at every time point is also known and stored in default. The philosophy of obtaining shaft torque, angular acceleration and angular speed values is given below.

5.8 **Compute Torque and Acceleration Values by Iteration.** In order to instantaneously compute the torque and acceleration of shaft at the time t_n, the known conditions at time t_{n-1} are considered. If the acceleration at this time point is $\dot{\omega}_{n-1}$, consider the driving torque and load torque T_L. We can now get two estimates of shaft torques

$$Ts_1 = T_D - J_D \dot{\omega}_{n-1}$$

$$Ts_2 = T_L + J_L \dot{\omega}_{n-1}$$

where T_D - Driving torque in Nm
T_L - Load torque in Nm
J_D - Inertia of driving component in kgm^2
J_L - Load inertia in kgm^2
Ts_1, Ts_2 - Estimated torque value acting on the both sides of shaft in Nm

Now these two are in effect equal and we can obtain a first approximation T_s by averaging

$$T_s = \frac{Ts_1 + Ts_2}{2} \qquad \ldots\ldots (5.2)$$

We can then obtain two values for acceleration $\dot{\omega}$, i.e. $\dot{\omega}_1$ and $\dot{\omega}_2$.

$$\dot{\omega}_1 = \frac{T_D - T_s}{J_D}$$

$$\dot{\omega}_2 = \frac{T_s - T_L}{J_L}$$

And an average of these can also be obtained:

$$\dot{\omega} = \frac{\dot{\omega}_1 + \dot{\omega}_2}{2} \qquad \ldots\ldots (5.3)$$

The process can be iterated until preset tolerances for T_s and $\dot{\omega}_1$ are reached.

5.9 **Transfer of inertias to shaft model.** Above iteration calculation in shaft model requires inertia values J_D and J_L from adjacent driving and load components. These values may be variable with respect to time and will be supplied through two signal links on the model. Figure 5.4 shows the free-body diagram showing torques acting on shaft and adjacent models and figure 5.5 shows a transfer of inertias to shaft model.

5.10 **Iterative tolerances.** The following tolerance proportional functions have been used to give reasonably rapid convergence, at the same time, preserving simulation accuracy.

1) Torque tolerance

$$\Delta T = \frac{T_{max}}{10000}$$

2) **Angular acceleration tolerance**

$$\Delta\dot{\omega} = \frac{\dot{\omega}_{max}}{100000}$$

5.11 **Obtain Shaft Angular Speed.** Once a value for shaft angular acceleration $\dot{\omega}$ is obtained, the value for shaft angular speed ω can be obtained by integration.

5.12 The shaft has been modelled using this procedure, and the model has been incorporated to the HASP model library with the mnemonic SHA1. A full description and documentation are given in Appendix B of this thesis.

TEST OF CIRCULAR SHAFT MODEL SHA1

5.13 In order to check the behaviour of the above model, a simple test simulation has been carried out. A test circuit and its computer link diagram is shown in figure 5.6. The load consisted of inertia, speed dependent friction and an external component T_{EXT} supplied by the duty cycle model DEDT (T_{EXT} is shown in figure 5.7). The angular speed change is shown in figure 5.8. The engine and load started from rest and accelerated to nearly 500 rev/min during the first second of computation. When the external load was applied the speed fell to 20 rev/min and recovered to about 270 rev/min when the load fell to 500 Nm (In practice this is not a realistic case as most engines would stall at speeds as low as 20 rev/min).

The simulation data provided for this simple test is shown in table 5.1.

PHILOSOPHY OF OBTAINING TORQUE, ANGULAR ACCELERATION AND ANGULAR SPEED OF A SHAFT-GEARBOX UNIT

5.14 **Introduction.** Consider a shaft-gearbox unit which comprises a gearbox and its input and output shafts, which is shown in figure 5.9. The flexibility, torque loss and inertia effects of input and output shafts are neglected. The same technique can be applied to obtain the torque, angular acceleration and angular speed of input shaft of gearbox, and then relevant values for output shaft of gearbox can be obtained by means of the relationships of variables between input and output shafts of a gear box as given below.

i) angular acceleration:

$$\dot{\omega} = G\ \dot{\omega}_L$$

where G - Gear box ratio

$\dot{\omega}$ - Angular acceleration of input shaft in $1/s^2$

$\dot{\omega}_L$ - Angular acceleration of output shaft in $1/s^2$

ii) angular speed:

$$\omega = G\ \omega_L$$

where ω - Angular speed of input shaft in 1/s

ω_L - Angular speed of output shaft in 1/s

iii) Equivalent value for the load torque of input shaft:

$$T_{Leq} = \frac{T_L}{G}$$

where T_{Leq} - Equivalent value for the load torque of input shaft in Nm

T_L - Load torque of output shaft in Nm

iv) Equivalent value for the load inertia of input shaft:

$$J_{Leq} = J_{G1} + \frac{J_L + J_{G2}}{G^2}$$

where J_{Leq} - Equivalent value for the load inertia of input shaft in kgm^2

J_L - Load inertia of output shaft in kgM^2

J_{G1} - first gear wheel inertia in kgM^2

J_{G2} - Second gear wheel inertia in kgM^2

5.15 **Iterative Procedure for Torque and Acceleration Values of Gearbox Input Shaft.** Supposing the conditions at time t_{n-1} are known, for instance, ω_{n-1} and $\dot{\omega}_{n-1}$, the torque and angular acceleration values of the gearbox input shaft can be computed instantaneously by the iterative procedure employed earlier (paragraph 5.8).

i) Estimate the torque acting on the shaft by using the driving torque T_D and load torque T_L. Two estimates of torque acting on the two sides of input shaft, shown in figure 5.10, are given by

$$Ts_1 = T_D - J_D \dot{\omega}_{n-1}$$

$$Ts_2 = T_{Leq} + J_{Leq} \dot{\omega}_{n-1}$$

where Ts_1, Ts_2 - Estimated torque acting on the both sides of input shaft in Nm

J_D - Inertia of driving component (eg. Engine or hydraulic motor) in kgM^2

T_D - Driving torque in Nm

We can get an average torque acting on the shaft

$$T_s = \frac{Ts_1 + Ts_2}{2} \qquad \ldots\ldots (5.4)$$

ii) We can then obtain two values for acceleration $\dot{\omega}$, i.e. $\dot{\omega}_1$ and $\dot{\omega}_2$.

$$\dot{\omega}_1 = \frac{T_D - T_s}{J_D}$$

$$\dot{\omega}_2 = \frac{T_s - T_{Leq}}{J_{Leq}}$$

And an average can be obtained by employing the equation:

$$\dot{\omega} = \frac{\dot{\omega}_1 + \dot{\omega}_2}{2} \qquad \ldots\ldots (5.5)$$

The process can be iterated until preset tolerances for T_s and $\dot{\omega}_1$ are reached. The tolerances $\Delta\dot{\omega}$ and ΔT are determined using the method described in paragraph 5.10.

5.16 **Solution for Angular Speed of Gear Box Input Shaft.** Once a value for shaft angular acceleration $\dot{\omega}$ is obtained, the value for shaft angular speed ω can be supplied by integration.

5.17 **Solution for Angular Speed of Gear Box Output Shaft.** According to the variable relationships between the input and output shafts of a gear box, the angular speed of the output shaft can be obtained as follows:

$$\omega_L = \frac{\omega}{G}$$

5.18 The shaft-gearbox-unit model has been modelled using this procedure, and the model has been incorporated to the HASP model library with the mnemonic SHA2. A full description and documentation are given in Appendix B of this thesis. The model was tested as before and the results were satisfactory.

SIMULATION OF AN ENGINE-HYDROSTATIC TRANSMISSION SYSTEM

5.19 **Circuit description.** Simulation of an engine hydrostatic transmission system was used as a practical example in order to assess the usefulness of the shaft model with the iterative procedure. The circuit under consideration includes two shafts which are used to transmit energy. One is linked with a diesel engine which is used as a prime mover and a fixed displacement pump delivering fluid to a hydraulic motor, and another is linked with a hydraulic motor and a rotary load including inertia in addition to viscous friction. The diesel engine drives the hydraulic pump via the No.1 shaft to supply fluid to the hydraulic motor and then the motor drives the rotary load via the No.2 shaft. An orifice restrictor is included in the pipeline connecting the pump and motor. When the rotary load including external load and inertia, friction etc. becomes extremely large, the pressure in the supplied line of the hydraulic circuit builds up to a high level and the relief valve opens. A circuit diagram for the engine hydrostatic transmission system is shown in figure 5.1.

5.20 **Theoretical analysis.** In order to simplify the analysis, the engine is assumed to be governed, and the engine will therefore operate on the governor line, the speed reducing slightly as the torque increases until the point of full fuelling is reached. After this the governor becomes inoperative and the engine speed drops off rapidly as torque increases. Maximum torque is reached at 600 rev/min.

i) engine output torque is given by:

a) if $\omega_E \geqslant 1350$ rev/min

$$T_E = 100 - (\omega_E - 1350)\frac{100}{150} \qquad \ldots\ldots (5.6)$$

b) if $\omega_E < 1350$ rev/min

$$T_E = 120 - (\omega_E - 600)\frac{(120-100)}{(1350-600)} \qquad \ldots\ldots (5.7)$$

c) if $T_E < T_{M0}$ $\quad T_E = T_{M0}$ $\qquad \ldots\ldots (5.8)$

T_E - diesel engine torque in Nm
T_{M0} - maximum negative output torque of engine in Nm
ω_E - angular speed of shaft in rev/min

This is modeled in PMR0.

ii) the shaft angular speeds are determined by the model SHA1 described earlier

iii) the pump flow rate and torque are given by model PU00 which has its salient equation:

$$Q_p = D_p\,\omega_p - C_p\,P_1 \qquad \ldots\ldots (5.9)$$

where D_p - pump displacement
ω_p - pump shaft angular speed
C_p - pump effective leakage coefficient

$$T_p = D_p\,(1 + C_{fp})\,P_1 + B_p\,\omega_p \qquad \ldots\ldots (5.10)$$

where B_p - pump viscous coefficient

C_{fp} - pump friction coefficient

iv) the rate of change of pressure in pipe 1 is given by model PI05

$$\frac{dP_1}{dt} = \frac{\beta}{V_1}\,(Q_p - Q_{or} - Q_r) \qquad \ldots\ldots (5.11)$$

where β - fluid bulk modulus

P_1 - fluid pressure in the pipe 1

V_1 - fluid volume in pipe 1

Q_{or} - flow rate through orifice restrictor

Q_r - flow rate through relief valve

Although the model also incorporates cavitation checks they were not involved during the pressure simulation.

v) the relief valve flow rate is given by model PC01:

$$Q_r = \frac{P_1 - P_c}{k} \qquad \ldots\ldots (5.12)$$

where P_c - cracking pressure of relief valve

k - flow coefficient of relief valve

vi) the orifice flow rate is given by model OR00:

$$Q_{or} = C_d A \left| \frac{2(P_1 - P_2)}{\rho} \right|^{0.5} = K(P_1 - P_2)^{0.5}$$

where C_d - flow coefficient
A - opening area of orifice
P_2 - fluid pressure in pipe 2
ρ - fluid density

The model linearises the equation about the condition $Q = 0$. About this point.

$$Q_{or} = k\ (P_1 - P_2) \qquad \ldots\ldots (5.13)$$

vii) the rate of change of pressure in pipe 2 also is modeled by PI05.

$$\frac{dP_2}{dt} = \frac{\beta}{V_2} (Q_{or} - Q_m) \qquad \ldots\ldots (5.14)$$

viii) the motor flow rate

$$Q_m = D_m\ \omega_m + C_m\ P_2 \qquad \ldots\ldots (5.15)$$

where D_m - motor displacement
ω_m - motor shaft angular speed
C_m - motor effective leakage coefficient

ix) the driving torque of motor

$$T_m = D_m\ (1 - C_f)\ P_2 - B_m\ \omega_m \qquad \ldots\ldots (5.16)$$

where C_f - motor friction coefficient

B_m - motor viscous coefficient

iix) rotary load torque of shaft

$$T_L = T_{EXT} + (T_f + f_v |\omega| + f_w \omega^2) sign(\omega) \quad \ldots\ldots (5.17)$$

where T_f - friction torque

$$T_f = \begin{cases} T_c & \text{if } |\omega| > 0 \\ T_s & \text{if } |\omega| = 0 \end{cases}$$

T_c - coulomb friction torque

T_s - stiction torque

T_{EXT} - external variable or constant load torque

f_v - viscous friction coefficient

f_w - windage coefficient

5.21 **Simulation data.** Figure 5.11 is a computer linking block diagram for the simulation of an engine hydrostatic transmission system. In this simulation, the rated torque and angular speed of the diesel engine model are 100 Nm and 1350 rev/min respectively. The system simulation is carried out respectively according to two different initial conditions of the engine which are given by

Case 1: $\omega_0 = 1500$ rev/min

Case 2: $\omega_0 = 1400$ rev/min

The parameters of other components used in the simulation is shown in table 5.2.

5.22 **Simulation results and discussion.** Since the starting performance of a diesel engine is not taken into account, different initial conditions chosen for the diesel engine will make different simulation results. Here two initial conditions are considered and the relevant system simulation results are discussed below respectively.

5.23 **Case 1:** Assume that the initial angular speed of the shaft No.1 is 1500 rev/min which is the zero torque speed for the diesel engine. In this case, the output torque of diesel engine is zero as can be seen from its torque characteristic shown in figure 5.12. The system load consists of inertia, speed dependent friction and an external component supplied by the duty cycle model DEDT shown in figure 5.13. The system pressure changes are shown in figure 5.14. The angular speed changes of shaft No.1 and No.2 are shown in figures 5.15 and 5.16 respectively. The engine starts from the initial speed 1500 rev/min and decelerates to nearly 1470 rev/min during the zero and two seconds of computation and falls to nearly 1390 rev/min when the external load is applied. The hydraulic motor starts from rest and goes up to 7.6 rev/min rapidly and then falls to 7.5 rev/min gradually and finally falls to about 6 rev/min when the external load increases to 10000 Nm from zero. The changes of engine torque and pump load torque are shown in figure 5.17. The changes of motor torque and system load torque are shown in figure 5.18.

Parameter	Sign	Value	Unit
engine fuel flow	q_f	6.2×10^{-5}	kg/s
engine fuel value	Q_f	4.24×10^{7}	kJ/kg
engine efficiency	η_e	0.4	
engine inertia	J_E	1	kgm^2
maximum torque acting on shaft (SHA1)	T_{max}	2×10^{3}	Nm
maximum angular acceleration	$\dot{\omega}_{max}$	1x10	$1/s^2$
initial angular speed	ω_0	0	rev/min
rotary load inertia	J_L	1	kgm^2
stiction torque	T_s	1x10	Nm
coulomb friction torque	T_c	8	Nm
viscous coefficient	f_w	$2.x10^{1}$	Nm/rev/min
windage loss coefficient	f_w	0	$Nm/rev/min^2$
1st stage external load	T_{ex1}	0	Nm
2nd stage external load	T_{ex2}	1×10^{3}	Nm
3rd stage external load	T_{ex3}	5×10^{2}	Nm

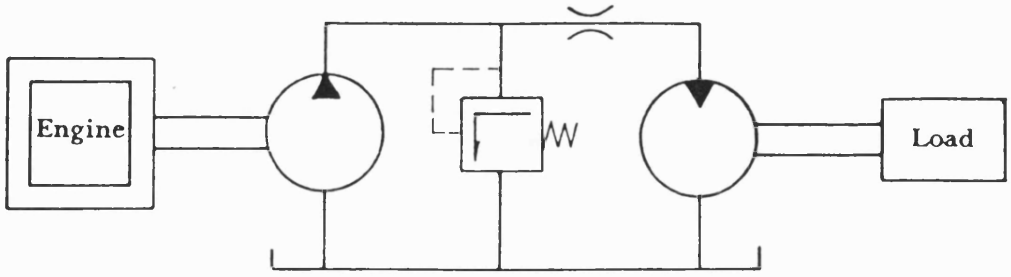
Table 5.1 Data values used for simulation of model SHA1

Parameter	Sign	Value	Unit
engine rated angular speed	ω_R	1.35×10^3	rev/min
rated torque	T_R	1×10^2	Nm
max output torque	T_{max}	1.2×10^2	Nm
min angular speed	ω_{min}	$6. \times 10^2$	rev/min
max negative output	T_{MO}	-2.5x10	Nm
maximum torque acting on shaft (SHA101)	T_{max}	2.5×10^2	Nm
maximum angular acceleration	$\dot{\omega}_{max}$	1×10^{-1}	$1/s^2$
initial angular speed	ω_0	1.5×10^3	rev/min
	or	1.4×10^3	rev/min
pump displacement	D_p	5×10^{-2}	l/rev
orifice diameter	d_o	0.75	mm
coefficient of discharge	C_d	0.69	
relief valve cracking pressure	P_c	1.1×10^2	bar
relief valve constant	k	0.5	(L/s)/bar

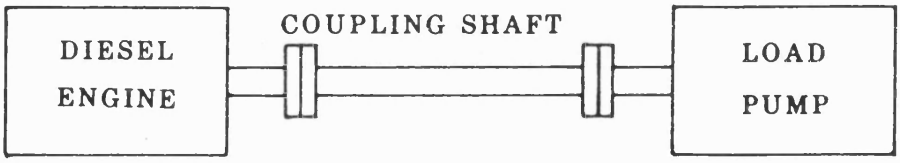
Table 5.2 Data values used for simulation of an engine hydrostatic transmission system

Parameter	Sign	Value	Unit
Pipe internal diameter	d	15	mm
pipe length	L	1.5	m
fluid/pipe Bulk Modulus	β	$7.x10^{3}$	bar
initial pressure	P_0	0	bar
motor displacement	X_m	$1.5x10^{-3}$	m^3/rad
total leakage loss coefficient	$K_{leakage}$	$2.08x10^{11}$	
motor inertia	J_M	1	kgm^2
motor effective viscous coef.	f_ω	$7.96x10^{2}$	Ns/m
motor friction coefficient	C_f	0.047	
maximum torque acting on shaft (SHA102)	T_{max}	$1.5x10^{4}$	Nm
maximum angular acceleration	$\dot{\omega}_{max}$	$1x10^{-1}$	m/s^2
initial angular speed	ω_0	0	rev/min
rotary load inertia	J_L	4	kgm^2
stiction torque	T_s	0	Nm
coulomb friction torque	T_c	0	Nm
viscous coefficient	f_v	$2.35x10^{3}$	Nm/rev/min
1st stage external load	T_{ex1}	0	Nm
2nd stage external load	T_{ex2}	$1x10^{4}$	Nm

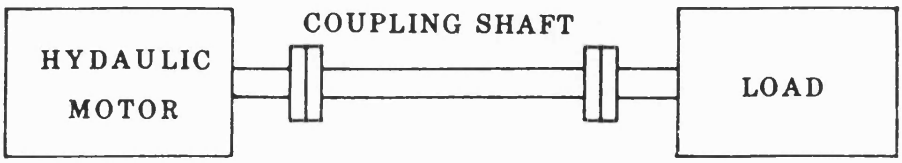
Table 5.3 Data values used for simulation of an engine hydrostatic transmission system



(a) Circuit diagram



(b) Power transmission sub-system I



(c) Power transmission sub-system II

Figure 5.1 Engine hydrostatic transmission system

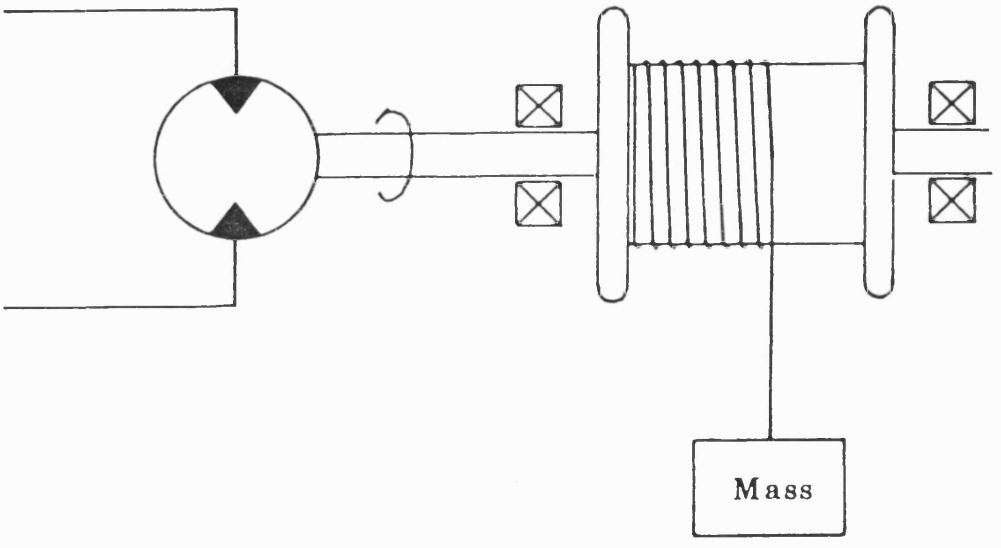


Figure 5.2 Schematic of hydraulic winch system

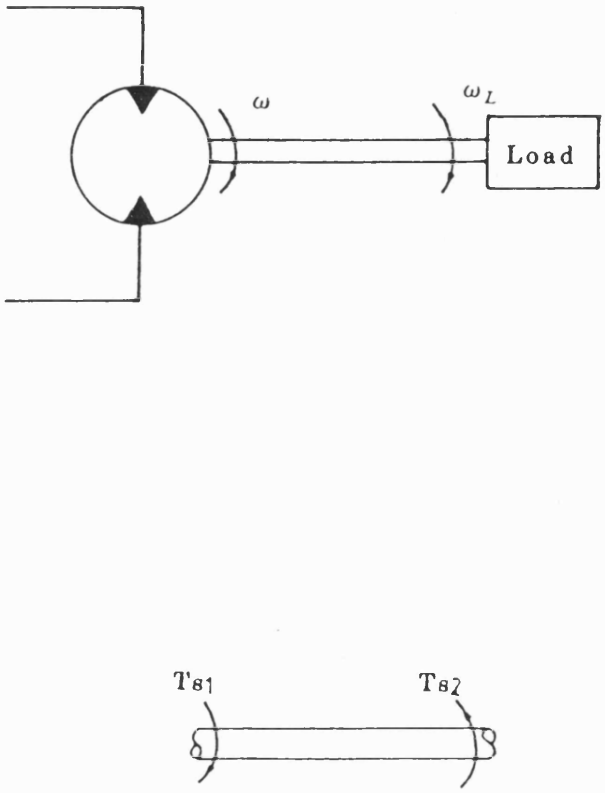


Figure 5.3 Motor-shaft-load subcircuit of a hydrostatic transmission system

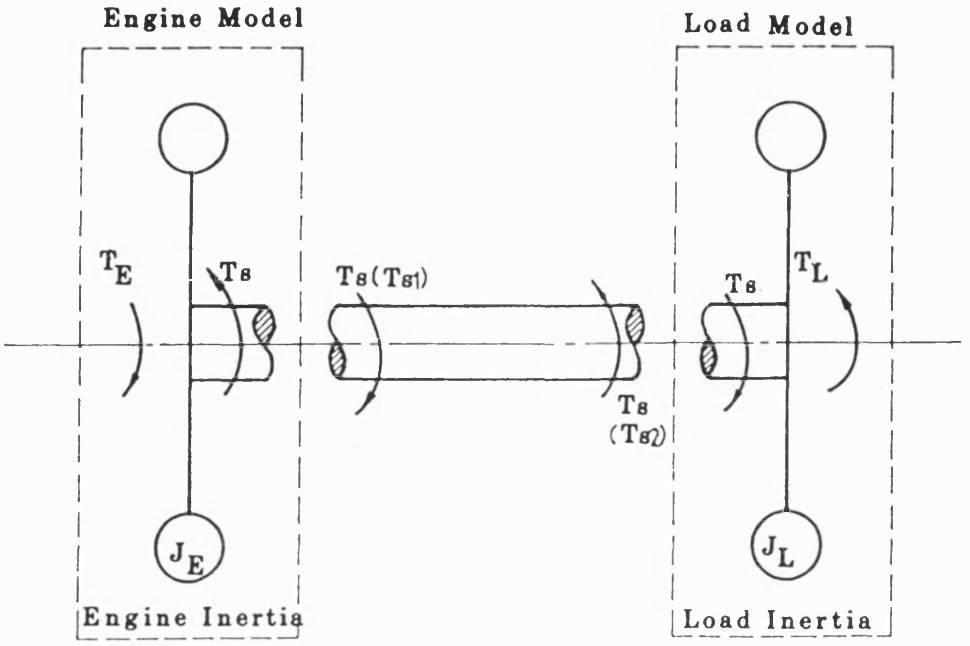


Figure 5.4 Free-body diagram showing torques acting on shaft and adjacent models

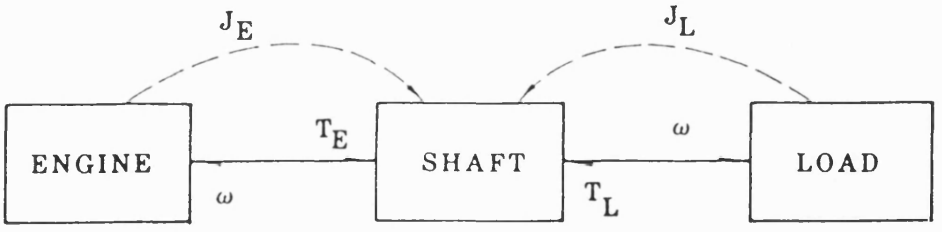
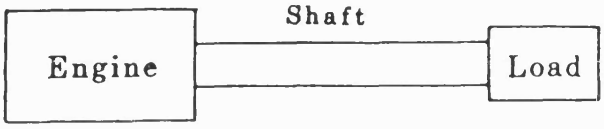


Figure 5.5 Transfer of inertia to shaft model



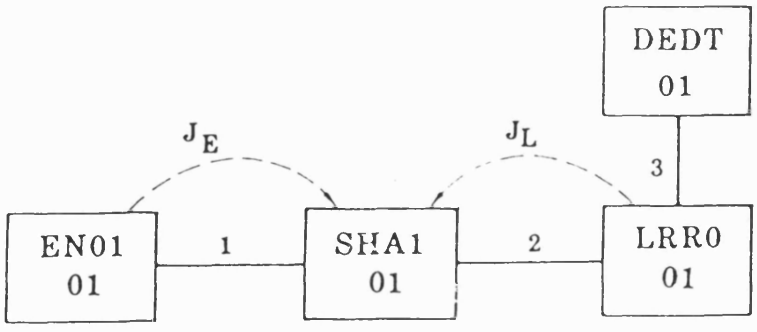


Figure 5.6 Test circuit and block diagram for simulation of shaft model

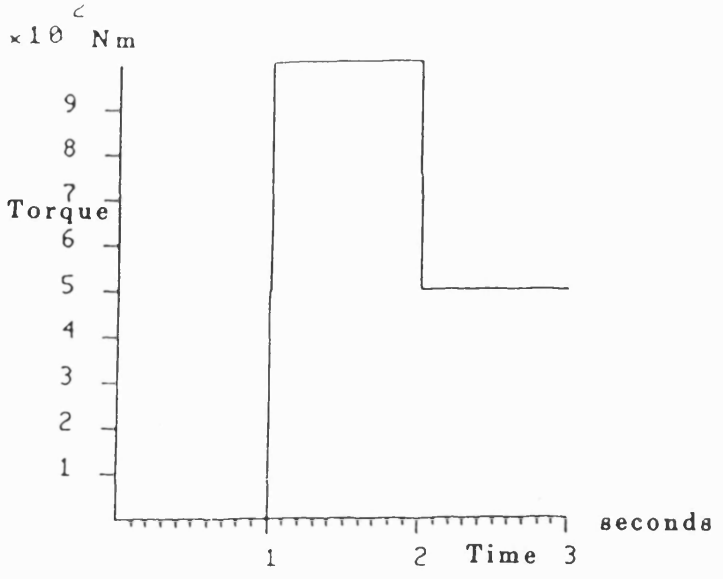


Figure 5.7 External load torque acting on rotary load model

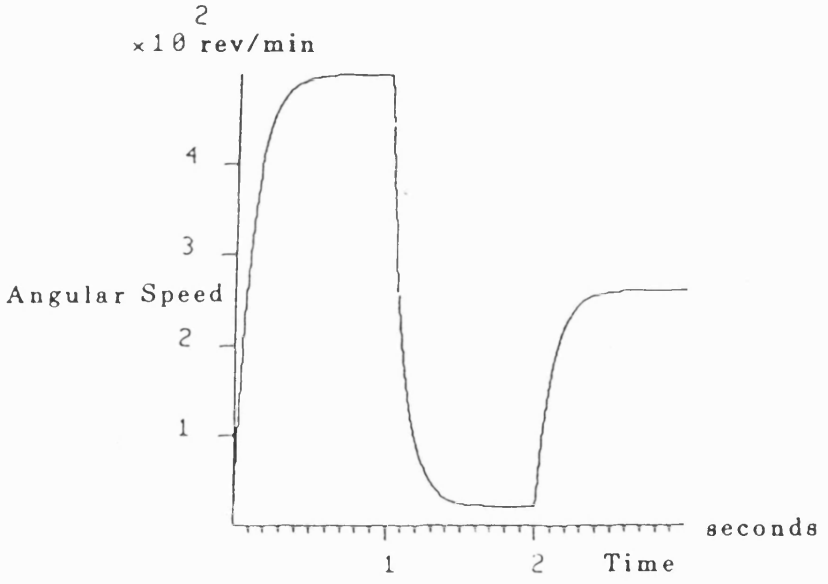


Figure 5.8 Shaft angular speed

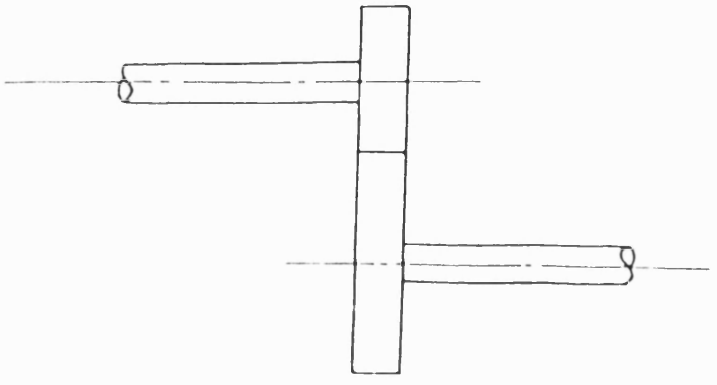


Figure 5.9 Schematic of a shaft-gearbox unit

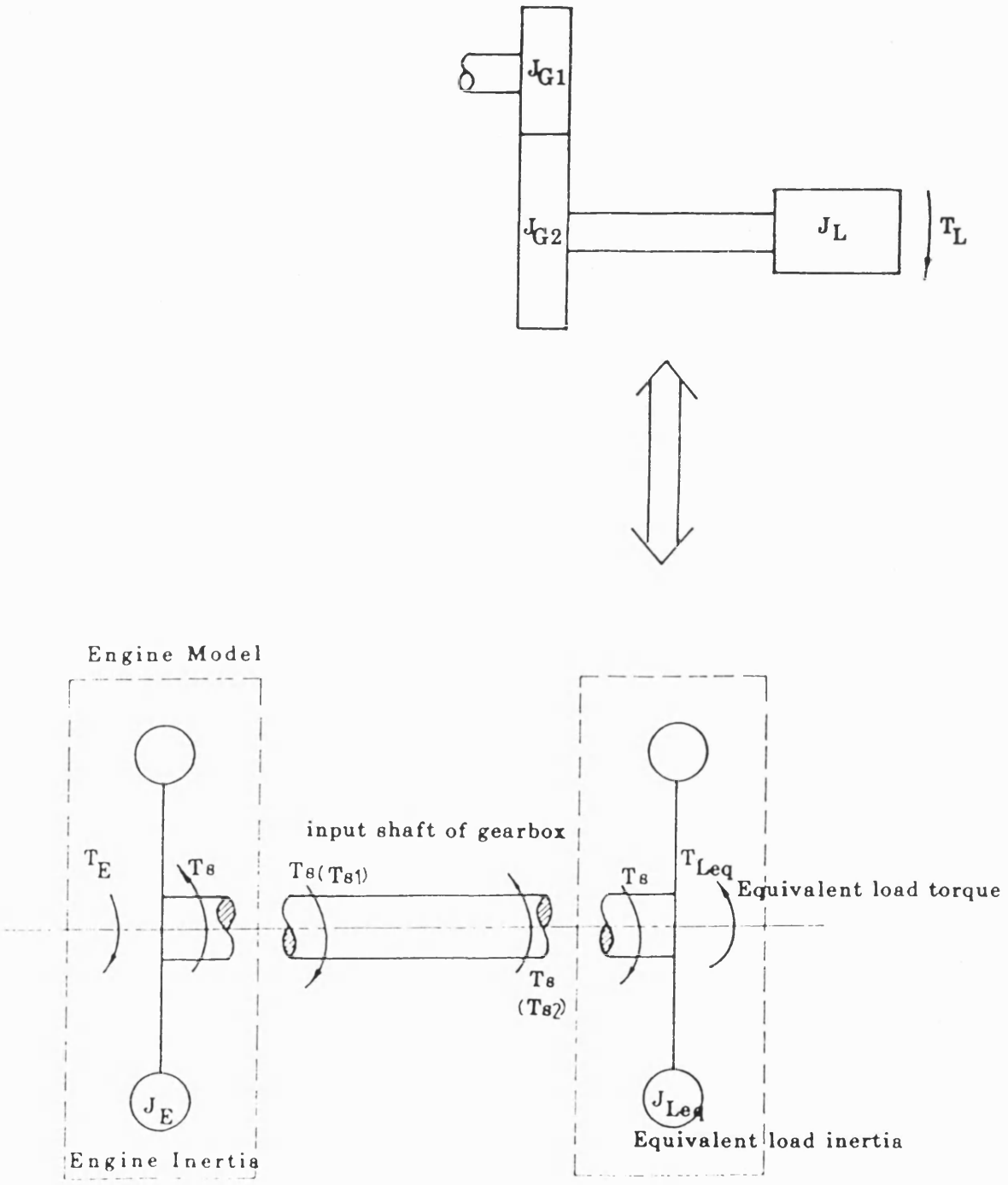


Figure 5.10 Free-body diagram showing torques acting on input shaft of a gearbox and engine, gearbox

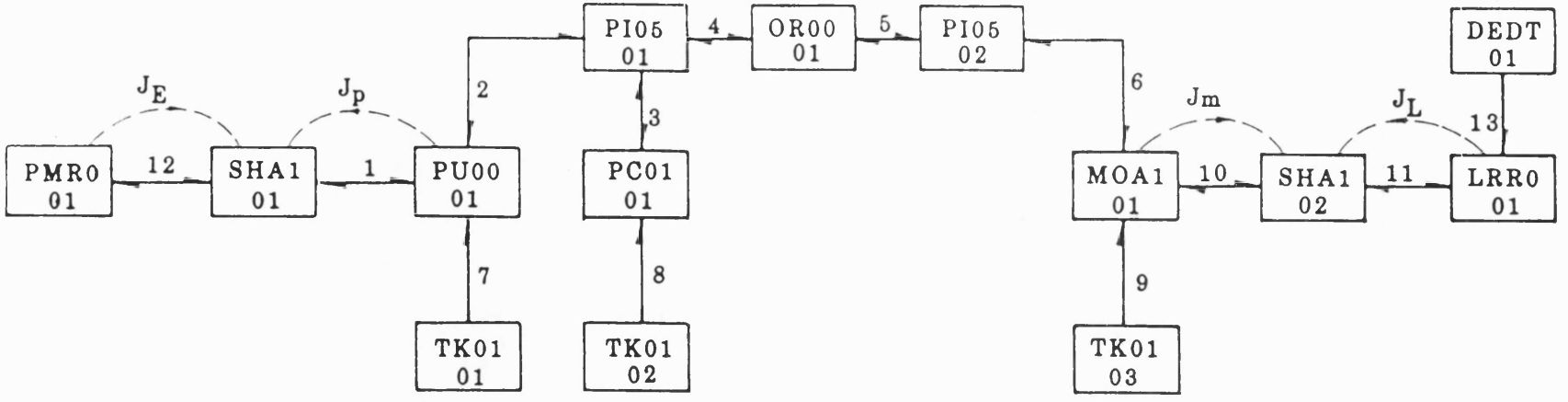


Figure 5.11 Computer block diagram of engine hydrostatic transmission system

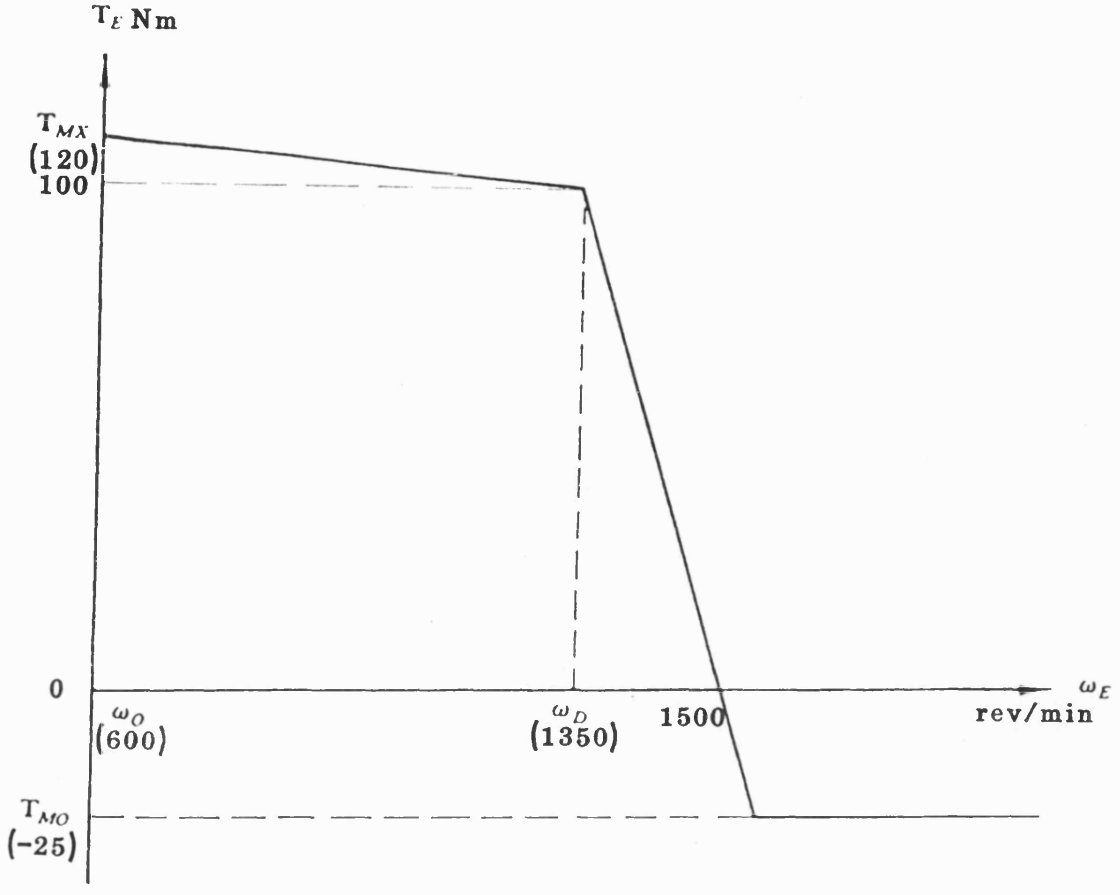


Figure 5.12 Torque/speed characteristics of a diesel engine

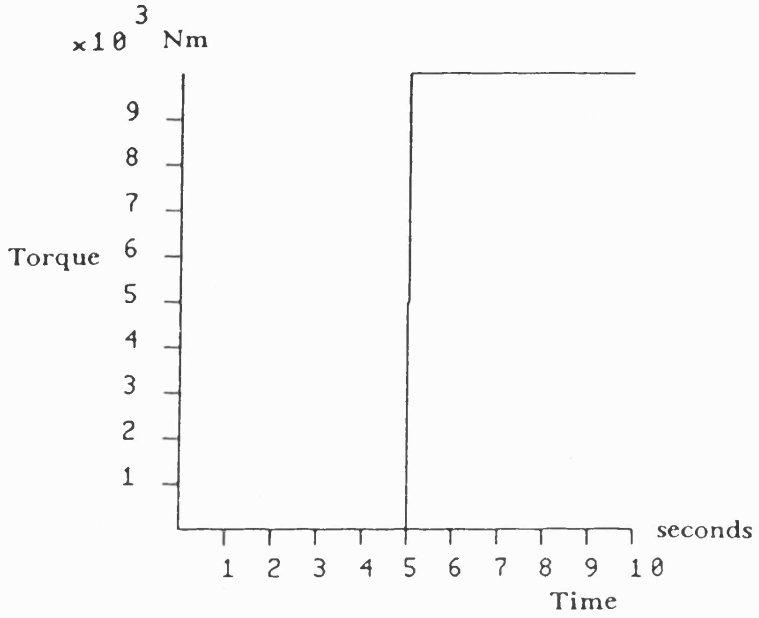


Figure 5.13 A variable external load torque acting on the engine hydrostatic transmission system

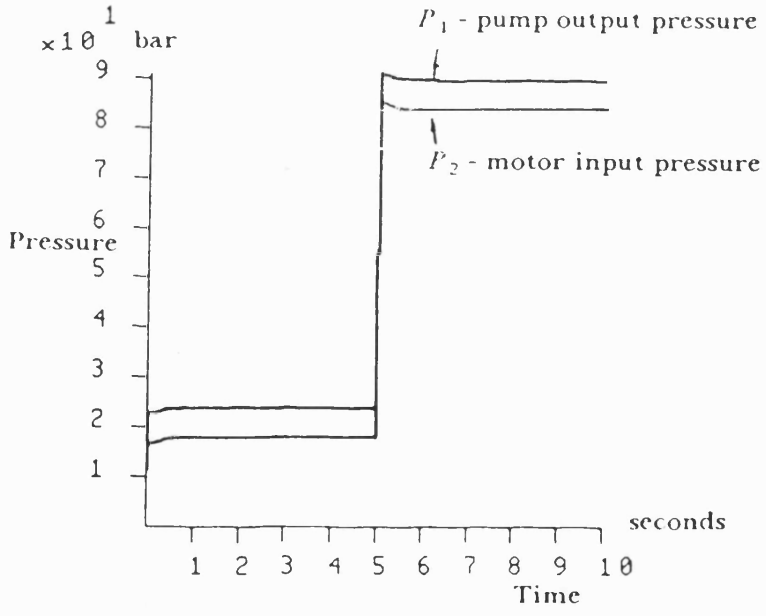


Figure 5.14 Pressures of hydrostatic transmission

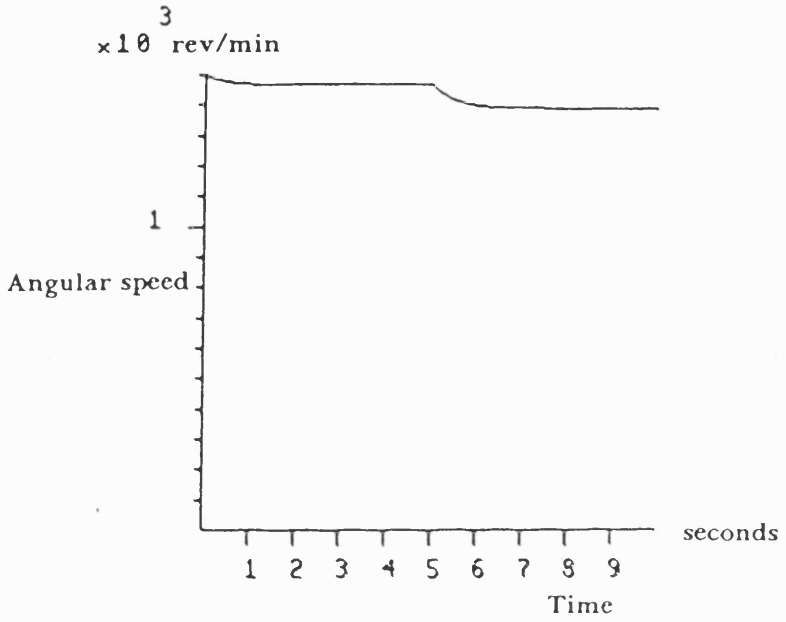


Figure 5.15 Angular speed of engine output shaft (case 1)

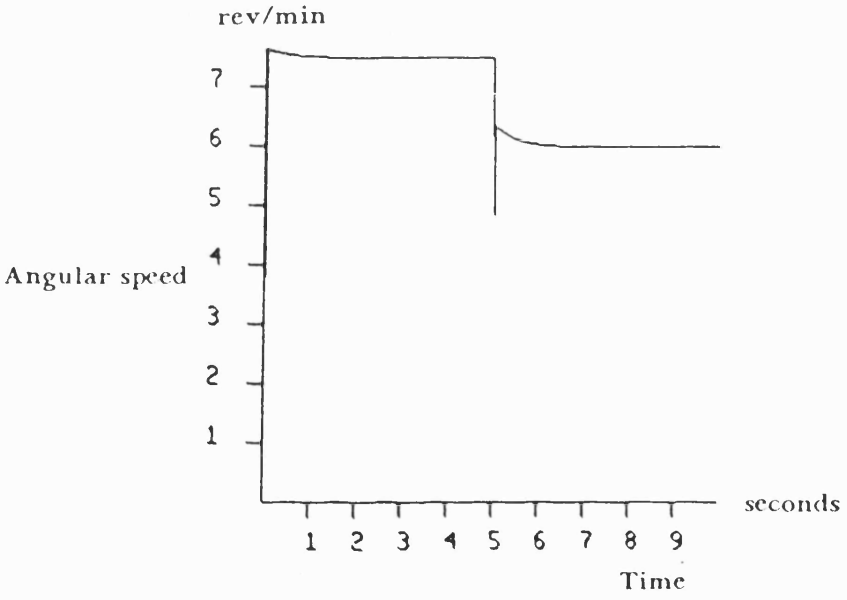


Figure 5.16 Angular speed of hydraulic motor output shaft

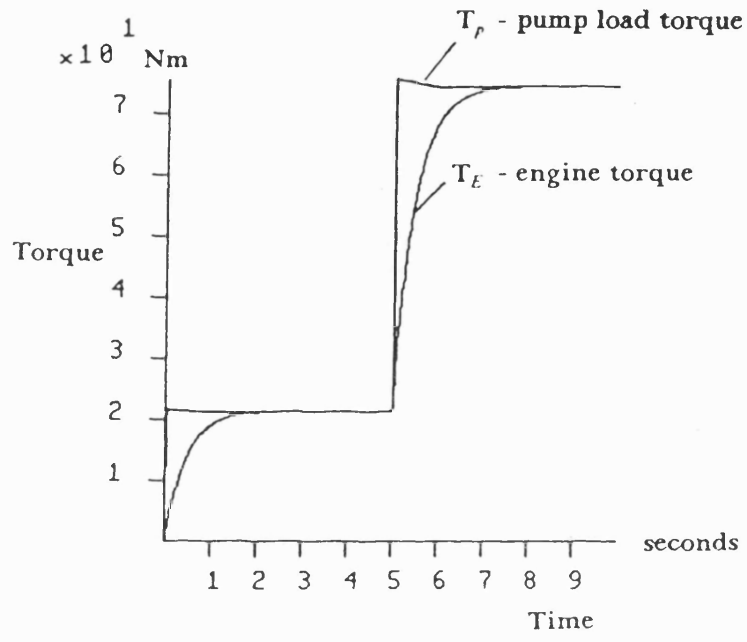


Figure 5.17 Engine output torque and pump load torque

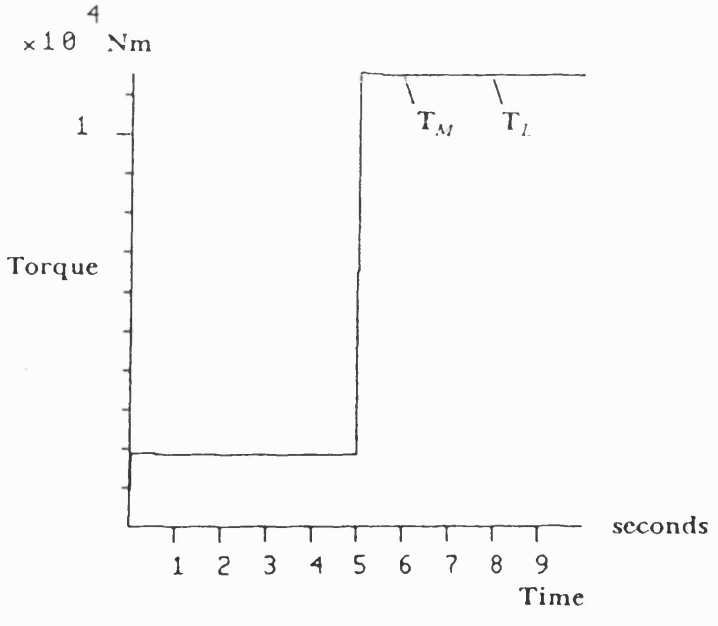


Figure 5.18 Hydraulic motor torque and rotary load torque

CHAPTER 6

SINGLE STEP NUMERICAL INTEGRATION METHODS FOR DISCONTINUOUS AND STIFF SYSTEMS

INTRODUCTION

6.1 The digital simulation of hydraulic systems is usually concerned with the numerical solution problem of a physical system described by ordinary differential equations with defined initial values. Each system to be modelled is ultimately arranged into a set of first order differential equations of the forms:

$$\frac{d\bar{y}}{dt} = f\,(t\,,\bar{y}\,) \quad \textit{with known} \quad \bar{y}_0 = y(0) \qquad \ldots\ldots (6.1)$$

where $\bar{y}$ is an array of size n, n represents the number of system state variables and $f\,(t\,,\bar{y}\,)$ is an array of size n.

Thus for a system with n state variables.

$$\begin{aligned}
\frac{dy_1}{dt} &= f_1(y_1, y_2, \ldots, y_n, t\,) \\
\frac{dy_2}{dt} &= f_2(y_1, y_2, \ldots, y_n, t\,) \qquad \ldots\ldots (6.2)\\
&\ldots\ldots\ldots\ldots \\
\frac{dy_n}{dt} &= f_n(y_1, y_2, \ldots, y_n, t\,)
\end{aligned}$$

It is required of an integration algorithm to progress the solution of the variables $y_1, y_2, \cdots y_n$ from the known values of time t to their values at a new time $t+dt$ where dt is a small integration time increment. Although these equations may be linear or non-linear, continuous or discontinuous and stiff or non-stiff, their numerical solutions always are obtained by one of either single-step numerical integration methods or multistep methods.

6.2 A multistep method requires information at previous time-steps such as $[t_{n-2}, t_{n-1}]$, $[t_{n-1}, t_n]$, $[t_n, t_{n+1}]$ and uses the previous values of the state variables and their derivatives at these steps to set up weighted functions to provide estimated values for the state variables at the next step. Examples of multistep methods are Gear's method and Adam's method [22][23]. Gear's method is particularly suitable for the numerical solution of non-linear and stiff systems, but is at a disadvantage when solving discontinuous systems (although in this case it can be improved by one order method or single step method with high order), and hence may be unsuitable for many engineering problems. It is also extremely complex, and difficult to understand which means that when failure of the technique occurs the reasons are difficult to find. Although this shortcoming could be overcome by using some special modelling techniques such as the cubic smoothing method, it still affects the application of this integration method due to the complex modelling process of physical systems.

6.3 A single step integration method, however, only involves information in a single step $[t_n, t_{n+1}]$ and uses known (initial or previous computed) information at time point t_n to provide the value y_{n+1} at time point t_{n+1}. Examples of single step methods are Euler's method and the Runge Kutta methods. These single step methods have an ability of solving discontinuity problems due to their self-starting property, moreover they also can cope with many of the mathematical stiffness problems encountered in the simulation of fluid power systems. Particularly, a variable step length single step method is suitable for the numerical solution of discontinuous and stiff systems, and a single step method with a fixed step length is suitable for discontinuous and weak stiff systems, but not for stiff systems, for which a very small integration time step and a correspondingly long simulation time would be required. The reasons about this were described by Rolfe[32], Lambert[26]. This chapter is mainly concerned with solving discontinuous and stiff problems using single step numerical integration methods, and in particular the case of a fourth order explicit Runge-Kutta method, a modified Runge-Kutta-Merson

method and a modified Runge-Kutta-Fehlberg method. Finally three relevant numerical integrators called RK4, KUTMER and KUTFEH have been adopted for the hydraulic automatic simulation package (HASP).

TYPICAL SINGLE-STEP NUMERICAL INTEGRATION METHODS

6.4 Consider the solution of the initial value problem

$$\frac{dy}{dt} = f(t, y) \qquad \ldots\ldots (6.3)$$

where y represents a set of solutions of a system described by i ordinary first order differential equations. The system described by the equation (6.3) is simple but is useful for describing following integration methods and their properties.

6.5 **Simple Euler Method.** This is the most basic method of all numerical integration algorithms and is expressed as:

$$y_{n+1} = y_n + h\ f(t_n, y_n) \qquad \ldots\ldots (6.4)$$

where $f(t_n, y_n)$ - the derivative of state variables at n th integration step.
h - the integration step size
y_{n+1} and y_n - the values of the state variable y corresponding to n th and $n+1$th integration steps.

This Euler equation chooses the slope value $f(t_n, y_n)$ at time point t_n as an average slope and then the solutions of differential equations are obtained by approximating the actual curves to a series of straight lines as shown in figure 6.1. Obviously this simple method suffers from the disadvantage that a very small step size is often required to obtain an accurate solution.

6.6 **Accuracy and stability of simple Euler method.** In the case of the simple Euler method, the error of numerical solution at each step can be found by direct comparison of the simple Euler formula to the Taylor's expansion which gives an exact solution if all terms are included[22],[19]. i.e.

Simple Euler: $y_{n+1} = y_n + h\ f_n$

Taylor's expansion:

$$y_{n+1} = y_n + h\ f_n + \frac{h^2}{2!} f'_n + \frac{h^3}{3!} f''_n + \cdots$$

From the comparison it can be seen that the error ϵ introduced by using the simple Euler formula is given by:

$$\epsilon = \frac{h^2}{2!} f'_n + \frac{h^3}{3!} f''_n + \cdots \qquad \ldots\ldots (6.5)$$

or it is approximated by:

$$\epsilon = \frac{h^2}{2!} f'_n = O\ (h^2) \qquad \ldots\ldots (6.6)$$

This error is termed the local truncation error of the method and is proportional to the square of the step size. However this is the order of error for one step only. The error over many steps becomes $O(h)$ and such accumulated error is termed the global error.

6.7 The stability characteristics of a numerical integration method can be assessed using the solution of the first order differential equation

$$\frac{dy}{dt} = \lambda y \qquad \ldots\ldots (6.7)$$

The differential equation has the solution:

$$y = A\ e^{\lambda t} \qquad \ldots\ldots (6.8)$$

where λ is known as an eigenvalue and this may be either real or complex. Applying simple Euler method to above equation we get:

$$y_{n+1} = y_n + h\ \lambda\ y_n \qquad \ldots\ldots (6.9)$$

The stability requirement is that any numerical error incurred does not grow as the solution progresses. If y_n is assumed to be exact and an initial error ϵ_n is introduced so that the solution can be represented by the parameter Z_n where

$$Z_n = y_n + \epsilon_n$$

Then $Z_{n+1} = Z_n + \lambda\ h\ Z_n \qquad \ldots\ldots (6.10)$

If equation (6.9) is subtracted from (6.10), then

$$\begin{aligned} \epsilon_{n+1} &= \epsilon_n + h\ \lambda \epsilon_n \\ &= (\ 1 + h\ \lambda\ )\ \epsilon_n \end{aligned} \qquad \ldots\ldots (6.11)$$

For stability we require that

$$|\ \epsilon_{n+1}\ | < |\ \epsilon_n\ | \qquad \ldots\ldots (6.12)$$

$$| \epsilon_n \ (1 + h \ \lambda) | < | \epsilon_n | \qquad \ldots\ldots (6.13)$$

Thus the stability condition of the simple Euler method is given by:

$$| 1 + h \ \lambda | < 1 \qquad \ldots\ldots (6.14)$$

The stability region for the simple Euler method is shown in figure 6.2. When $h \ \lambda$ lies inside the circle, the solution obtained using simple Euler method will be stable, outside the circle the solution will be unstable.

6.8 **Improved Euler method.** The principle of the improved Euler method is that the simple Euler method is applied twice in sequence as shown in figure 6.3. First of all the simple Euler is used to obtain a predicted value for y'_{n+1} at t_{n+1} using the slope $f\ (y_n)$

$$y'_{n+1} = y_n + h \ f\ (t_n, y_n) \qquad \ldots\ldots (6.15)$$

The value for y'_{n+1} is then used to obtain the predicted slope $f\ (y'_{n+1})$ at t_{n+1} point. An improved estimate for the value of y_{n+1} is then found using the average of the two slopes:

$$y_{n+1} = y_n + \frac{h \ [f\ (t_n, y_n) + f\ (t_{n+1}, y'_{n+1})]}{2} \qquad \ldots\ldots (6.16)$$

An analysis of accuracy[19] shows that the error of one step of the improved Euler method is $O(h^3)$ (local error) and an accumulation of errors from step to step is $O(h^2)$ (global error). Obviously the solution obtained using improved Euler method is more accurate than that using simple Euler method.

6.9 **Backward Euler Method.** With Backward Euler method the value of the function y_{n+1} at t_{n+1} over a step-length is determined by:

$$y_{n+1} = y_n + h\ f\ (t_{n+1}, y_{n+1}) \qquad \ldots\ldots (6.17)$$

Here it can be seen that in order to calculate y_{n+1} it is necessary to calculate $f\ (t_{n+1}, y_{n+1})$ which requires y_{n+1} to be known. Backward Euler is therefore implicit and has to be solved using some kind of iteration procedure. The representation of Backward Euler method is shown in figure 6.4.

6.10 The simplest iteration approach is to first predict, using simple Euler, the value of the function y_{n+1} at t_{n+1}.

for predictor, $y^p_{n+1} = y_n + hf\ (t_n, y_n)$ $\qquad \ldots\ldots (6.18)$

The slope $f\ (y^p_{n+1})$ at t_{n+1} is then estimated using the value y^p_{n+1}, and then it is used to re-estimate, or correct, the value for y_{n+1}

for corrector, $y^c_{n+1} = y_n + hf\ (t_{n+1}, y^p_{n+1})$ $\qquad \ldots\ldots (6.19)$

The new slope $f\ (t_{n+1}, y^c_{n+1})$ is then found and this process is repeated for either a fixed number of iterations or until the difference between successive values for y^c_{n+1} is within a set tolerance.

6.11 A stability analysis of the Backward Euler method gives a region of stability as shown in figure 6.5, i.e.

$$|\ 1 - h\ \lambda\ | > 1 \qquad \ldots\ldots (6.20)$$

provided that the method is corrected to convergence.

The numerical solution of equation (6.3) is stable if the modulus of $(1 + h\ \lambda)$ is larger than the unity. Otherwise, the solution becomes unstable.

6.12 The Backward Euler method is only first order accurate and has a local error which is equal to that of simple Euler method[22],[26]. In most cases the step sizes will be fixed by the desired accuracy for the solution of simulation. Therefore very small step-lengths will be required if Backward Euler method is used to obtain more accurate solutions.

6.13 **Fourth Order Explicit Runge-Kutta Method.** A further advance in efficiency (i.e. obtaining the most accuracy per unit of computational effort) can be secured with a group of methods due to the German mathematicians Runge and Kutta. The fourth order Runge-Kutta methods are widely used in computer solutions to differential equations and these methods recognise that the change in "Y" during the step-length "h" does not necessarily occur at a constant rate. The formula for a fourth order explicit method is[22]:

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \qquad \ldots\ldots (6.21)$$

The term $\frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$ is a weighted mean average of the gradient of the function over the step-length h which matches Taylor's series up to order 4. A graphical representation of the method is given in figure 6.6.

The term K_1 is the derivative of the function y_n at position n.

$$K_1 = f(t_n, y_n) \qquad \ldots\ldots (6.22)$$

Slope K_1 is used to obtain an estimate of the function half-way across the interval h, $\frac{y_{n+1}}{2} = y_n + K_1 \frac{h}{2}$. This function is then used to obtain an estimate of the slope at $(\frac{n+1}{2})$ and is given by:

$$K_2 = f\,(t_n + h/2, y_n + K_1 \frac{h}{2}) \qquad \ldots\ldots (6.23)$$

K_2 is then used at n to obtain a re-estimate of the function at $(\frac{n+1}{2})$ giving $y_{\frac{n+1}{2}} = y_n + K_2 \frac{h}{2}$. The slope at the revised value $y_{\frac{n+1}{2}}$ gives K_3:

$$K_3 = f\,(t_n + h/2, y_n + K_2 \frac{h}{2}) \qquad \ldots\ldots (6.24)$$

K_3 is then used to estimate the function at the total interval h giving $y_{n+1} = y_n + K_3 h$. y_{n+1} is used to obtain another estimate of the function gradient K_4 for the full interval h. This gives

$$K_4 = f\,(t_n + h, y_n + K_3 h) \qquad \ldots\ldots (6.25)$$

All of the information necessary to compute K_1, K_2, K_3 and K_4 is always available at every time-step including the first.

6.14 **Accuracy and stability.** Suppose the present time step size is h, the local error of fourth order explicit Runge-Kutta method is the term in the Taylor's series expansion of order 5. And then assuming terms of higher order than 5 are insignificant, the local error of solution is:

$$y_{n+1} - y\,(t_{n+1}) = C_5 h^5 \qquad \ldots\ldots (6.26)$$

where C_5 is the coefficient for the term in Taylor's series and is related with the value of $y\,(t)$ in the interval $[t_n, t_{n+1}]$. Therefore the local error term for the fourth order Runge-Kutta is $O(h^5)$; the global error would be about $O(h^4)$. It is usually computationally more efficient than the improved Euler method because, while four evaluations of the derivative function are required per step rather than two, the steps can be many-fold larger for the same accuracy. The stability region of the

fourth order Runge-Kutta method is shown in figure 6.7.

6.15 **Advantages of method.** The fourth order explicit Runge-Kutta algorithm is a single-step method with high order and self-starting properties. Its main advantages are:

(i) The solutions of state variables at time t_{n+1} only depend on state variables between t_n and t_{n+1}, rather than information at several previous time points (t_n ,t_{n-1}, and t_{n-2} etc.), This is a main reason why the discontinuity problems are easy to solve in a single-step method.

(ii) Computational accuracy is high due to the order of the method.

(iii) The stability region is much wider than other low order Runge-Kutta methods, which can be seen in figure 6.7.

(iv) Many stiffness problems in the simulation of fluid power systems could be solved by the 4th order explicit Runge-Kutta method although a long computation time might be required.

(v) Discontinuity problems could be solved by using "IF STATEMENT" technique. The operation of this technique is controlled by a logical indicator "LIMIT", both in the component model and integrator.

6.16 **Disadvantages.** The main disadvantages and limitations of this fixed step method are:

(i) Since the stability region is not wide enough, the simulation of some systems with serious stiffness problems necessitates a very small integration step size.

(ii) The choice of integration time step for a fixed step method is rarely optimal.

6.17 In order to overcome above disadvantages of the fixed step-length fourth order Runge-Kutta method, variable step-length Runge-Kutta method and other higher-order Runge-Kutta formulas such as Runge-Kutta-Merson method and

Runge-Kutta-Fehlberg method have been developed and can be used to advantage in determining a suitable size, h.

6.18 **Variable Step 4th Order Runge-Kutta Method.** The Above fixed step-length fourth order Runge-Kutta method can be modified as a variable step-length method when an error test is introduced to control the integration step size. According to the principle of the error estimate approach called *Richardson Extrapolation* [23], an error estimate for step size control can be expressed by:

$$ERROR = | y_{n+1} - y^*_{n+1} | \qquad \ldots\ldots (6.27)$$

where the first estimated value y_{n+1} to the exact value $y(t_{n+1})$ at t_{n+1} is obtained from the equation (6.21) by evaluating the derivative values at four points in an integration step interval $[t_n, t_{n+1}]$. The second estimated value y^*_{n+1} is obtained by evaluating the derivative values in the same interval in which the fourth order RK method is used twice. In this case, three derivative values need to be evaluated for obtaining $y_{n+1/2}$ at $t_n + \frac{h}{2}$ point because the $f(t_n, y_n)$ value at t_n is ready. Four derivative values need to be evaluated for obtaining y^*_{n+1} at $t_{n+1/2} + h/2$. Therefore this error computation for step control will take **5.5** derivative evaluations in each step and a representation for the estimation of y_{n+1} and y^*_{n+1} is shown in figure 6.8.

6.19 **Runge-Kutta-Merson Method.** The Runge-Kutta-Merson (or call Kutta-Merson) algorithm is a fourth order Runge-Kutta process that simultaneously gives an approximation to the single step error. The cost of this is one additional derivative function evaluation. Thus the determination of the first approximation y_{n+1} of $y(t_{n+1})$ takes **five** derivative function evaluations. This method is used for solving $\frac{dy}{dt} = f(t, y)$ given t_n, y_n. The calculations are carried out using following equations given by Merson (1957)[22],[33].

$$k_1 = h\ f(t_n, y_n),$$

$$k_2 = h\ f(t_n + h/3, y_n + k_1/3),$$

$$k_3 = h\ f\ (t_n + h/3, y_n + k_1/6 + k_2/6).$$

$$k_4 = h\ f\ (t_n + h/2, y_n + k_1/8 + 3k_3/8).$$

$$k_5 = h\ f\ (t_n + h, y_n + k_1/2 - 3k_3/2 + 2k_4).$$

$$y_{n+1} = y_n + \frac{k_1 + 4k_4 + k_5}{6} + O(h^5); \qquad \ldots\ldots (6.28)$$

$$y_{n+1}^{(*)} = y_n \frac{k_1 - 3k_3 + 4k_4}{2} + O(h^5) \qquad \ldots\ldots (6.29)$$

The error estimation for controlling the integration step size is given by:

$$| E | = \frac{| y_{n+1} - y_{n+1}^{(*)} |}{5} \qquad \ldots\ldots (6.30)$$

The right hand side of equation (6.30) can be used to control the step size. Since this method only takes **five** derivative function evaluations rather than **5.5**, Kutta-Merson method is more efficient and faster than variable step length Runge-Kutta method as developed by Richardson. The error estimation method shown in equation (6.30) has been used successfully in a number of automatic step selection methods.

The stability region of Kutta-Merson method is slightly bigger than that of fourth order Runge-Kutta method.

6.20 **Runge-Kutta-Fehlberg Method.** Another variable step Runge Kutta process is the fifth order Runge-Kutta-Fehlberg method which is described by:

$$y_{n+1} = y_n + \left(\frac{25k_1}{216} + \frac{1408k_3}{2565} + \frac{2197k_4}{4104} - \frac{k_5}{5} \right); \qquad \ldots\ldots (6.31)$$

The error estimation for controlling the integration step size is given by:

$$E = \left(\frac{k_1}{360} - \frac{128k_3}{4275} - \frac{2197k_4}{75240} + \frac{k_5}{50} + \frac{2k_6}{55} \right) * 0.02 \qquad \ldots\ldots (6.32)$$

where:

$$k_1 = h\ f\ (t_n, y_n).$$

$$k_2 = h\ f\ (t_n + \frac{h}{4}, y_n + \frac{k_1}{4}).$$

$$k_3 = h\ f\ (t_n + \frac{3h}{8}, y_n + \frac{3k_1}{32} + \frac{9k_2}{32}).$$

$$k_4 = h\ f\ (t_n + \frac{12h}{13}, y_n + \frac{1932k_1}{2197} - \frac{7200k_2}{2197} + \frac{7296k_3}{2197}).$$

$$k_5 = h\ f\ (t_n + h, y_n + \frac{439k_1}{216} - 8k_2 + \frac{3680k_3}{513} - \frac{845k_4}{4104}).$$

$$k_6 = h\ f\ (t_n + \frac{h}{2}, y_n - \frac{8k_1}{27} + 2k_2 - \frac{3544k_3}{2565} + \frac{1859k_4}{4104} - \frac{11k_5}{40});$$

The basis for the Runge-Kutta-Fehlberg method is to compute two Runge-Kutta estimates for the new value of y_{n+1} for h, we compare the estimates of y_{n+1} using fourth- and fifth-order Runge-Kutta formulas and comparing the results. Moreover, both equations make use of the same $k's$, so only **six** derivative function evaluations are needed. In addition, the changes of the step size h are controlled by the value of error calculated for that fourth oreder and fifth order methods. The error estimate for the fifth order method is given above by equation (6.32). The computational work for y_{n+1} is smaller than fourth order variable step-length Runge-Kutta method but larger slightly than Runge-Kutta-Merson method at each step. However, since the local error of the Runge-Kutta-Fehlberg method with variable step is $O(h^6)$, a larger integration step size could be chosen under a same tolerance size, and hence the solution using this method should be most accurate and the simulation CPU time spent is small compared with other two variable step-

length methods mentioned above. Therefore it seems appropriate to employ this method for solving discontinuous and stiff differential equations.

The stability region of fifth order Runge-Kutta-Fehlberg method is same as that of fifth order Runge-Kutta method, which is shown in figure 6.9.

6.21 **Assessment of typical single step numerical methods.** We have discussed the merit and deficiencies of seven typical single step numerical methods in preceding paragraphs. The advantages of these single step algorithms are:

i) these algorithms are self-starting;

ii) the step size h can be changed without affecting the use of special algorithms;

iii) That for many problems the use of a small enough integration step h will give an accurate and stable result.

The disadvantages of the algorithms are:

i) it is difficult to select the proper value of h ;

ii) the number of stages v (number of derivative function evaluations) required per step may result in excessive computing time.

6.22 According to above analysis of seven typical single-step methods, the fourth order explicit Runge-Kutta method, the variable step Runge-Kutta-Merson method and the fifth order Runge-Kutta-Fehlberg method are highly recommended as suitable integration procedures for the digital simulation of fluid power systems. They are self-starting, easy to program for a computer and require only a moderate amount of computer storage. Since the solution at each step needs only information at last time point, the discontinuous problems in component models can be overcome using simple IF STATEMENT modelling technique. The fourth order explicit Runge-Kutta method is mainly suitable for the non-stiff or weak stiff systems because it is a fixed step method. It is shown later that special modelling technique in combination with the fourth order Runge-Kutta-Merson method and

fifth order Runge-Kutta-Fehlberg method allow the simulation of potentially stiff problems. Very stiff problems or some physical violations caused by non-state variables could force that the integration step size becomes very small, even near zero. As a result, the estimation of state variables could go to infinite loop because the step size becomes smaller and smaller. In order to solve this kind of problem, above integration methods have been modified by means of the simulation requirement of engineering systems and then three new integrators were developed by the author for the HASP simulation package and are called as:

1) RK4 - fourth order explicit Runge-Kutta integration routine.

2) KUTMER - fourth order variable step-length Runge-Kutta-Merson integration routine.

3) KUTFEH - fifth order variable step-length Runge-Kutta-Fehlberg integration routine.

The explanation and testing of these integrators will be given later.

PHILOSOPHY FOR SOLVING STIFF AND DISCONTINUOUS PROBLEMS USING SINGLE STEP INTEGRATION ALGORITHMS

6.23 **Discontinuity Difficulties in Gear's method.** It is known that the best numerical integration methods for solving stiff problems are "multistep predictor-corrector methods". Gear's method is a predictor-corrector type of multistep integration method for solving stiff systems and it has been employed by HASP for many years[6],[7],[8]. The stability region of Gear's method with orders 1 to 6 is shown in figure 6.10. The integration accuracy and computational efficiency depends on the complexity and stiffness of the problem to be solved.

6.24 However, since Gear's method uses the trends in the solutions of the differential equations at past values to predict the solutions at future values

employing predictor-corrector polynomials, the discontinuities in the models of hydraulic systems can cause integration failure. This is because the solution depends on the predictor-corrector polynominals of Gear's method which are the result of previous information (state variables and their derivatives). Suppose the correct solution of a state variable y at whole time range is shown in figure 6.11. If the previous values of this state variable at x_{n-3}, x_{n-2}, x_{n-1}, x_n points are known, the solution of y at point $x_n + h$ depends on the polynominal $L(x)$ formed by y_{n-3}, y_{n-2}, y_{n-1}, y_n. Obviously the correct solution of y at $x_{n+1} = x_n + h$ could not be obtained by this polynominal. In other words, Gear's method or other multistep integration methods are only suitable for the integration of continuous systems. However, most models describing actual hydraulic systems and components are discontinuous, and hence special precautions and a cubic smoothing technique must be used in the modelling of systems or components when Gear's method is used to solve problems that are discontinuous in nature.

6.25 In order to integrate across the discontinuities using Gear's method, the technique used in HASP demands that the discontinuous functions in Function routine (system model segment AUX.FOR) are integrated piecewise in smooth sections by the inclusion of interval halving and restarting procedures. The discontinuous functions, or functions with discontinuous derivatives are divided into several regions in which discontinuous functions are smoothed as continuous function by using cubic smoothing technique. In this case, during whole integration process the mathematical equations used for the integration are continuous. This sort of technique ensures that the integration algorithm always operates within a continuous region, a detail about cubic smoothing method is described by Tomlinson[15]. In the mathematical modelling of hydraulic components, however, in order to ensure the continuity of equations, sometimes excessive cubic smoothing functions are required. Moreover for some component models with end-stop problems, dummy forces such as end-stop spring force and end-stop damping force are used to form the cubic smoothing functions. Since these dummy forces are not real end-stop reaction forces, the numerical solution of simulation could have a large error when the end-stop is hit. This has been shown in the simulation of a linear actuator in figure 1.6. On the other hand, model writers have to spend a lot of time on the computer modelling of components, and the coding of some component models seems to be too complex and difficult to understand. Obviously it is inconvenient for users who want to extend their own special purpose models.

A representation of solving discontinuity problems for current Gear's integration

routine in HASP is shown in figure 6.12.

6.26 **Philosophy for Solving Discontinuous Problems.** For the single step integration methods, the solution of state variables at any time point t_{n+1} only depends on previous information (state variables and their derivatives) at last time point t_n, and the solution at t_{n+1} is independent with intermediate calculation values between two time points. Therefore the effect of discontinuities in model equations on integration results can be erased when conditional statements for discontinuity problems are introduced into model calculation subroutines. For instance, consider the discontinuities in the model of a two stage relief valve. When a poppet hits a stop, the displacement, velocity of the poppet should become zero. In practical working condition these real restrained conditions of state variables can be expressed in conditional statements form by:

for the main poppet:

```
IF(X.LT.0.D0) THEN
  X=0.D0
  Vx=0.D0
  VXDOT=0.D0
END IF
```

for the pilot poppet:

```
IF(Y.LT.0.D0) THEN
  Y=0.D0
  Vy=0.D0
  VYDOT=0.D0
END IF
```

It is very important, however, that all restrained conditions of discontinuities in model equations have to be put into the conditional statement section of the model subroutines and they must be defined and written by the model writer. In the HASP simulation package, the operation of the conditional statement section is controlled by the integrator indicator "LIMIT".

6.27 **Function of LIMIT variable for discontinuous models.** The LIMIT variable whose value is supplied by the integrator is used to indicate whether an integration step has been completed or not. The "LIMIT" variable enables the model subroutines (in AUX) to communicate with the integration subroutine(RK4, KUTMER or KUTFEH). If LIMIT=0, that means that the model subroutines are called for the initialisation of state variables and relevant derivatives. If "LIMIT" is set to 1, the model subroutines are called for the computation of the estimated derivatives. In this case, the IF STATEMENTS describing discontinuities in model subroutines are not available for solving discontinuous problems because at this time the state variables are only intermediate values rather than the completed ones. Once the evaluation of derivatives is finished, a GOTO statement leads to RETURN statement and then the program execution goes back to the main section of the integration subroutine in which new state variables will be evaluated. If "LIMIT" is set to 2, that means an integration step has been completed and the new state variables have been stored in the array Y(I). In this case all of state variables must be checked to make sure whether they are suitable for the physically constrained conditions of the system. If some of them are not, they have to be assigned by the correct values defined by IF STATEMENTS.

6.28 Since the precise instant at which the discontinuities encountered will not be known, the conditional statement section describing the restrained conditions of discontinuities should be put after the mathematical equations in a model calculation subroutine.

6.29 **Integration step and solution accuracy.** According to above analysis, if the solution of some state variable at t_{n+1} is not suitable for the physically constrained condition, it has to be assigned a correct value defined by a IF STATEMENT. The solution of this state variable at this point is sure to be correct, but the accuracy of value on a straight line between two points t_n and t_{n+1} would depend on the step size. For instance, using a fixed step method such as the fourth order Runge-Kutta method, the solution of a state variable at current time point t_{n+1} depends on the value of the state variable at last time point t_n and the step size used. Thus a large error of the value on the straight line between two points could appear when a larger step h is used, and the representation of the error for the value on a straight line between two points is shown in figure 6.13. However, for the variable step size single step methods such as Runge-Kutta-Merson and Runge-Kutta-Fehlberg methods, the integration step h is controlled by the error estimate

formula. If the estimate of a state variable satisfies the error test, the time step is accepted or doubled, otherwise the time step is halved and the evaluation of state variables is carried out using a new step $\frac{h}{2}$. This process repeats until the error test is satisfied and then the estimates of state variables are accepted. For this reason, a reasonable step size h for discontinuous problems will be controlled by the error test of the estimate of state variables so that the error value on the straight line between two points y_n and y_{n+1} becomes as small as possible.

For instance, when the piston of an actuator moves to end of its stroke, the piston stops. In this case, the displacement $X = X_{max}$, velocity $v = 0$ and acceleration $a = 0$. An end stop model of this actuator can be expressed by:

$$X = X_{max} \text{ and } v = 0,\ a = 0; \quad \text{if } X > X_{max} \qquad \ldots\ldots (6.33)$$

and the relevant computer model is given by:

```
IF(X.GT.XMAX) THEN
  X=XMAX
  V=0.D0
  VDOT=0.D0
END IF
```

Figure 6.14 shows a representation of this sort of discontinuity. In the model calculation subroutine of the actuator, the displacement X and velocity v are state variables. Suppose a larger integration step h is used across a discontinuity point t_d, the estimates of state variables X and v won't satisfy the error test due to the possible physical violation caused by discontinuity, and then the integration step h is halved to reevaluate the estimates of X and v. Eventually the satisfactory X and v values which satisfy error test could be obtained when a smaller integration step h is employed. Since the step-length across discontinuous point is small, the accuracy of the value on the straight line between y_n and y_{n+1} should be high enough.

6.30 **Discontinuity difficulty using standard variable step size single step methods.** If the physical violation caused by discontinuities is very serious, the error test for the estimates of state variables could not be satisfied although the integration step h is halved again and again. In this case the evaluation of estimates of state variables could go to an infinite loop at some time point because the integration time step could become smaller and smaller. For example, the physical violation caused by velocity discontinuity shown in figure 6.14 makes that the estimate of state variable v at time point $t_n + h$ won't satisfy the error test although the integration step h has been reduced to extremely small.

6.31 **Solving method proposed.** In fact, the estimates of state variables at $t_n + h$ point can be corrected to satisfy the requirements of engineering systems using IF STATEMENTS technique. The solving method proposed here is to set a minimum integration step h_{min}, if the integration step h of the integrator reduces to this minimum value or smaller than it due to discontinuity violation or mathematical stiffness, we force the execution of estimates of state variables to jump out from the error test, ie. the estimates of state variables can be considered to be acceptable at time point $t_n + h_{min}$. In this case, LIMIT=2, these estimates of state variables will be checked whether they are suitable for the physical constrained conditions of the real system. If not, they will be assigned the correct values by the conditional statements in the components. But the choice of this minimum integration step h_{min} mainly depends on the requirement of stiffness systems and it will be discussed in paragraph 6.37.

6.32 **Philosophy of Solving Stiffness Problems.** The basic difficulty in the numerical solution of stiff systems is the satisfaction of the requirement of stability. Since the stability regions of explicit RK methods are small, a very small integration step would be required in order to obtain the correct solution of state variables in the stiff region. Therefore the basic principle of solving stiffness problems using single step methods is to ensure the stability of solutions of a system using adequate small integration step-length. However, if a fixed step-length single step method is used to simulate a stiff system, a very long CPU time would be required. If a variable step-length method is used, the CPU time spent for simulating a stiff system can be reduced considerably. In order to avoid a too large accumulated error caused by many small steps, normally we chose high order single step methods for solving stiff systems. In the HASP simulation package, two efficient single step methods, fourth order Runge-Kutta-Merson and fifth order

Runge-Kutta-Fehlberg methods have been employed for stiff and discontinuous systems, and relevant integration step can be automatically adjusted due to the stiffness change of the system to be investigated. If the system is very stiff, the integration step will be reduced to a very small size. Otherwise a large integration step required by stable solutions will be chosen.

6.33 **Effects of non-state variables on system stability.** The stability of solutions of a set of variable coefficient differential equations will vary with the coefficients of equations. For instance, in the simulation of a linear actuator system (see paragraph 6.53), the solution stability of state variables would be affected by a DVC (directional control valve) controller model DEDT whose output is a nonlinear non-state variable. This is because the model of the linear actuator system with DCV is a set of variable coefficient and nonlinear differential equations due to a nonlinear variable input from controller model. When the displacement input of DCV controller model varies rapidly[8], for instance, from zero position to fully open position, the nonlinear change of coefficients of differential equations for hydraulic pipes might affect the stability of solution considerably, nevertheless this kind of stiffness effect caused by a non-state variable could not be checked out in the error test of estimated state variables of integrator. Therefore it is possible that the system simulation using some models such as DCV controller model will fail.

6.34 **Pseudo-state variable method.** In order to overcome above difficulty, the output variable X supplied by the DCV controller model (duty cycle model) can be assumed to be a pseudo-state variable whose derivative is zero. i.e.

$$\frac{dX}{dt} = 0.0$$

Since $\frac{dX}{dt} = 0$, X value supplied from integrator is always equal to instantaneous output value of the controller model DEDT. However the X also is a state variable and can join the error test for the estimates of state variables during the integration process, and hence the physical violation caused by X stiff changes will be checked out by the error test and then the integration step is halved automatically. Eventually a reasonable small integration step chosen by the error test will ensure that the stable solution of the system being investigated is obtained. After transient

responses of state variables tend to be stable, of course, a reasonable large step-length will be automatically recovered by the integrator. This sort of difficulty was overcome in Gears integrator by employing LIMIT=3 in the stiff change region of current HASP duty cycle models, and a representation about this is shown in figure 6.15.

6.35 **Simulation test for two types of duty cycle models.** Consider a linear actuator system whose circuit diagram is shown in figure 6.16 and the computer linking diagram is shown in figure 6.17. A controller is used to control the position of a directional control valve in the circuit. If a HASP standard duty cycle model DE01 is used to simulate the controller, the simulation of this system using single step algorithms (KUTMER or KUTFEH) will fail due to the effects of step changes of non-state variable X (position of directional control valve) and relevant simulation results are shown in figure 6.18. However if using a modified model DER1 in which the output variable X is set as a pseudo-state variable (i.e. only one statement is inserted into the model DE01), this problem will be overcome because the stiffness effect caused by X is worked out and the integration step h can be halved automatically if necessary by means of the error tests of state variables in KUTMER or KUTFEH integration subroutine. Figure 6.19 shows correct simulation results.

6.36 **Modification of Standard Integration Methods.** Since the physical violation caused by mathematical discontinuity and stiffness might force that the integration step h becomes smaller and smaller before the completed values of state variables at a new time point are obtained, the evaluation for the estimates of state variables may become an infinite loop, and then the execution of the system simulation could stop there forever. In order to overcome this difficulty, therefore, a simple modification is completed in Runge-Kutta-Merson and Runge-Kutta-Fehlberg methods. i.e. assume a minimum integration step h_{min} is known, if the error test of estimates of state variables at t_{n+1} point

$$|\,ERROR\,| > TOL \qquad \ldots\ldots (6.34)$$

the integration step is halved:

$$h = \frac{h}{2}$$

According to this new step, the estimates of state variables are reevaluated and then are tested again using above error test formula. This process continues until the error test is satisfied. The modification proposed here is that if integration step is less than the minimum step h_{min}, the error test for estimates of state variables is finished at t_{n+1} point and the estimates of state variables are considered to be acceptable. This is defined as an integration step criterion given below.

if $h < h_{min}$, jump out from error test

if $h \geqslant h_{min}$, carry on error test

where h_{min} is used to control whether the error test for estimates of state variables is necessary or not.

6.37 **Simulation test for a minimum integration step size.** In order to consider the stability of general solutions, the minimum integration step size should be very small. However a too small h_{min} would take long long time to obtain the solution of state variables. For this reason, several tests for the choice of h_{min} were done using the linear actuator circuit shown in figure 6.16 and then a more reasonable minimum integration step h_{min} was chosen as $1x\ 10^{-6}$. Figure 6.20 shows the simulation results for the choice of h_{min}.

6.38 **Necessary Information in Model Attributes File COMPON.DAT for Using a Component Model with "If Statements".** As we mentioned earlier, the correction of discontinuous values of state variables is carried out in the model calculation subroutine by using IF STATEMENTS. The modified values of state variables of a component model must be transferred to Y(I) array in the system model segment AUX.FOR from link variables (EFF(N) or FLO(N) array) in order to supply correct state variables and their derivatives to the integrator. Therefore the state variable values Yi of the model must be reset in AUX by the values in relevant link variables EFF(i) and FLO(i). This can be done automatically by the program generator after the value at the end of second line of the model attribute section in COMPON.DAT file is set to 2.

The file COMPON.DAT includes such information as link and signal details as well as the numbers of state variables, real and integer parameters etc of a model. For instance, for a two stage relief valve dynamic model PCR1 the required attribute details (two lines) in COMPON.DAT is of the form:

```
PCR1
223054103N2
```

The second line shows that the model PCR1 has two external links, three internal links, five state variables, forty-one real constant parameters, three integer constants and no time (T) is required in the argument list of the model subroutines. It also shows that five state variables of PCR1 need to be reset at the bottom of the system model subroutine AUX.FOR because the value at the end of second line is set to 2. A basic structure of the AUX segment in which the model PCR1 is called is given below.

```
C
C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C %%%% SUBROUTINE AUX - CALLS UP MODEL CALCULATION SUBROUTINES
C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE AUX(DOT,Y,T,N,LIMIT)
            :
            :
C
C %%%% ASSIGN THE STATE VARIABLES CALCULATED BY THE INTEGRATOR
C %%%% TO THE APPROPRIATE LINKS
            :
      EFF( 4)=Y( 1)
      FLO( 4)=Y( 2)
      EFF( 5)=Y( 3)
      FLO( 5)=Y( 4)
      EFF( 6)=Y( 5)
            :
C
C %%%% CALL EACH MODEL CALCULATION SUBROUTINE IN THE ORDER
C %%%% DEFINED BY THE CALLING SEQUENCE DETERMINED IN THE
C %%%% PROGRAM GENERATOR SEGMENT PGCMP
            :
            :
      CALL PCR1( ...., EFF(4),FLO(4),EFF(5),FLO(5),EFF(6),FLO(6),
     +    LIMIT,CON3,ICON3,DOT(1),DOT(2),DOT(3),DOT(4),DOT(5))
            :
            :
C %%%% STATE VARIABLES ARE RESET BY THE VALUES IN EFF AND FLO ARRAYS
            :
      Y( 1)=EFF( 4)
      Y( 2)=FLO( 4)
      Y( 3)=EFF( 5)
      Y( 4)=FLO( 5)
      Y( 5)=EFF( 6)
            :
      END
```

IMPLEMENTATION OF FOURTH ORDER RUNGE-KUTTA METHOD

FOR THE HASP SIMULATION PACKAGE

6.39 **Structure of Fourth Order Runge-Kutta Method in HASP.** Fourth order explicit Runge-Kutta integration method, described in paragraph 6.13, has been implemented in the HASP simulation package and is currently being used for the numerical integration of state variables. Depending on the simulation step size h and total simulation time T_{end}, all the solutions of the state variables of a system at all time points of the simulation interval $[t_o, t_{end}]$ can be solved before the RETURN occurs. The structure of Runge-Kutta method in HASP is shown in figure 6.21, and consists of the following four parts:

(i) **Initialisation.** The function of this section is to initialise the derivatives of state variables. The subroutine AUX is called for this purpose.

(ii) **Subroutine AUX.** This defines all model functions being integrated. It also enables algebraic information to be exchanged between component models, derivatives of state variables to be transmitted from models to the integrator and state variables to be transmitted from the integrator to models requiring them. During the integration, equations including those relating to the discontinuous functions or conditions, are identified and controlled by an integrator status indicator "LIMIT" which can be altered in the integration subroutine. The subroutine AUX is called for three purposes, namely initialisation, computation of derivatives or predicted derivatives, and for the output of results. Figure 6.22 shows the interaction among the integration subroutine, AUX subroutine and the component model subroutines.

(iii) **Integration Loop** The evaluation and determination of state variables and their derivatives are made in an integration loop, in which the procedure of the evaluation and solution for the state variables and the derivatives are included.

(iv) **Subroutine OUT** An output subroutine OUT is automatically produced by the HASP program generator. This subroutine is called to store the results in a data file called CADRES.DAT. This file enables simulation results to be plotted or printed as required.

Runge-Kutta method integration subroutine named as RK4 in the HASP main segment MAIN.FOR is called for the integration of the state variables and a logical block diagram of the integration process is shown in figure 6.23.

6.40 **General comments for RK4 integrator.** Fourth order explicit Runge-Kutta method described in paragraph 6.13 is coded using Fortran 77 language. Flow chart for the coding is shown in figure 6.24. General comments for this integration routine called RK4 is given as follows:

1) The integration subroutine is called for the full duration of a simulation. The initial part of the subroutine sets up certain logical and other variables which are used and changed as the simulation progresses.

2) In the integrator coding RK4, the state variables of system equations are represented by the array variable Y(I). Similarly, the derivatives of the state variables are represented by the array variable DY(I). Moreover the array variables YC(I) and Y1(I) are the intermediate computational values of state variables and AA(I) are the coefficients of the iterative expression for the integration. Here *I* ranges from 1 to the maximum number of equations.

3)The values of the terms N, T, TEND, TAB as defined in Appendix D and the initial values of the state variables are supplied by calling the HASP subroutine CONTRL. The purpose of calling subroutine AUX is to calculate the derivatives of state variables so that the new state variables can be evaluated and finally determined. Subroutine OUT is used to store the simulation results. Both AUX, OUT are automatically written by the HASP Program Generator.

4) Label 140 in the subroutine is where the integration starts, and it is also the label where each new integration step commences when an integration step has completed successfully.

6.41 **Function of LIMIT Variable in Integrator.** LIMIT is the integrator status indicator and also is the logical control indicator in model subroutines. As a one-step method, the RK4 integrator only needs the stored values of state variables at the current time in order to solve new state variables at next time point, and hence it is necessary to investigate the solution status of state variables and their derivatives. Further if we know exactly the place where an integration step has been completed, it is possible to use the techniques described earlier to solve discontinuity problems such as end stop simulation in a hydraulic linear actuator. Otherwise the integrator is unaware of the possibility of violating physical conditions as may occur if an actuator passes through an end stop or a pressure drops below absolute zero. The method used to cope with a problem of this nature is to make use of an integer flag variable LIMIT. The integer flag variable LIMIT takes three possible values which depend on the state of the integration step. At the beginning of a simulation, it is set to 0. During the intermediate calculation process in the simulation, LIMIT is set to 1 and at the completion of a step it is set to 2. In the case, the model writer can use the technique of "IF STATEMENT" to cope with the discontinuity problems encountered in mathematical equations of model calculation subroutines.

The listing of RK4 integration subroutine is shown in Appendix D of this thesis.

IMPLEMENTATION OF MODIFIED RUNGE-KUTTA-MERSON METHOD FOR THE HASP SIMULATION PACKAGE

6.42 **Modified Runge-Kutta-Merson Method.** Basing on the standard Runge-Kutta-Merson method discussed in paragraph 6.19, two modifications were made in its computer coding of this method respectively:

a) add an error test for determining logical variables.

b) use a minimum integration step to control whether the error test for estimates of state variables is needed or not.

6.43 **Error test of determining logical variables B1 and B2.** This test was proposed by Barker (1969). Logical variables are used to control the estimation of state variables and to aid the option of integration step size. Before the evaluation of functions starts, two logical variables B1 and B2 will be determined as True or False and the relevant error test formula proposed by Barker is given as follows:

$$set:\ B_1 = False;\ B_2 = False$$

$$| ELOG | = \frac{(T + \Delta T) - (X_0 + H)}{\Delta T} \qquad \ldots\ldots (6.35)$$

$B_1 = TRUE$ and $B_2 = FALSE$: if $| ELOG | < -10^{-6}$

$B_2 = TRUE$ and $B_1 = FALSE$: if $-10^{-6} = | ELOG | \leqslant 10^{-6}$

Otherwise

$B_1 = FALSE$ and $B_2 = FALSE$: if $| ELOG | > 10^{-6}$

Once the evaluation of functions at each step size h is completed and the error estimate of state variables

$$ERROR \leqslant TOL$$

two logical variables B1 and B2 are used to determine whether the estimates of state variables are acceptable or not. If B1 or B2 is TRUE, these estimates are accepted and a new simulation time T is assigned by

$$T = T + H$$

then the simulation continues until the whole simulation time is reached. If both B1

and B2 are FALSE, another error test for estimates of state variables still needs to be done in order to judge if the current integration step size will be doubled. This error test is given by:

$$\text{if } | ERROR | > \frac{TOL}{64}, \quad H = H$$

$$\text{if } 0 < | ERROR | < \frac{TOL}{64}, \quad H = 2H$$

6.44 **Control of error test by using minimum step size h_{min}.** In order to avoid that the estimation of state variables goes to infinite loop due to the smaller and smaller step size, a minimum integration step size for the error test is proposed to be known in the Runge-Kutta Merson algorithm. In the simulation process, once the integration step size is less than this value, the error test for the estimates of state variables will be passed and those estimates of state variables are considered to be acceptable. The procedure about this is given by:

$$\text{if } | ERROR | = | 0.2 \, x \, (y_{n+1} - y^{*}_{n+1}) | > TOL \qquad \ldots\ldots (6.36)$$

the integration step is halved:

$$h = \frac{h}{2}$$

if $h < h_{min}$ jump out from error test

if $h \geqslant h_{min}$ carry on error test

Otherwise this error test continues until $| ERROR | \leqslant TOL$. The operation function of these two modifications in the integration subroutine can be seen clearly in the flow chart of Kutta-Merson's integration routine which is shown in figure 6.25.

6.45 **Main structure of KUTMER integrator.** The modified Runge-Kutta-Merson method mentioned above was implemented in the HASP simulation package and called as KUTMER integrator. The main structure of KUTMER integrator is shown in figure 6.26 and the function of various parts in this figure is described briefly below.

a) **Initialisation.** Initial information of state variables is required before the KUTMER integrator can operate. The function of this part is to initialise the derivatives of state variables using initial values of state variables. The model subroutine AUX is called for this purpose. In this stage, the integration status indicator LIMIT=0.

b) **Subroutine AUX.** This defines all component model subroutines being integrated. In the equations including discontinuous functions, the calculation of equations is controlled by the integration status indicator LIMIT. When the evaluation of state variables is finished at each step, LIMIT is set to be 2. In this case, the discontinuous function section described by IF STATEMENT in individual component model is used to check if the state variables satisfy the physical constrained conditions. If not, the values of state variables will be corrected by IF STATEMENTS. The AUX subroutine is called for three purposes, namely initialisation, evaluation for derivatives of state variables and for the output of simulation results.

c) **Error tests.** The KUTMER integrator is a variable step size integration routine. The control of the integration step size and the acceptance of estimates of state variables depend on two error tests employed in KUTMER integrator. One is for the logical control of chosing step size and another is for the numerical error control of chosing step size.

d) **Integration loop.** In the integration loop, the evaluation and determination of state variables and their derivatives are carried out and mathematical model subroutine AUX is called five times at each time step h. The operation of this integration loop is related with above two error

tests.

e) **Output interpolation.** In order to output the simulation results at every print point, the calculation values which are not exactly at print time point are used to establish a interpolation function and then the simulation results at each exact print time point can be obtained approximately.

f) **Subroutine OUT** This subroutine is called to store the simulation results of the system being investigated into a special data file called CADRES.DAT which enables results stored to be plotted or printed on the computer screen.

6.46 **General comments for KUTMER integrator.** The modified Runge-Kutta-Merson method is implemented in the HASP simulation package and is coded using Fortran 77 language. The listing of KUTMER integration subroutine is shown in Appendix D of this thesis. General comments for this integrator are given as follows:

1) The subroutine is called for the full duration of a simulation. The initial section of this subroutine sets up certain variables which are used and changed as the simulation progresses, and the maximum and minimum step size employed in the subroutine.

2) Label 140 in the subroutine is where the integration starts, and it is also the label where each new integration step commences when an integration step has completed successfully.

3) Label 120 is the label where the integration loop commences when the integration step size is h. Label 120 is also executed if step size h is halved or doubled.

4) After label 120 an error test is used for the choice of the logical control variables B1 and B2. The labels 101 to 205 in the subroutine are used to evaluate the estimates of state variables and their derivatives.

5) In the do loop (206) another error test is used to control the step size. If the error of estimates of state variables is larger than a set tolerance(TOL), the step size is halved and then return back to label 102 to re-evaluate the estimates again. If the integration step size is less than a set minimum step size (Hmin), the execution of function evaluation jumps to label 207. i.e. the estimates of state variables are considered to be acceptable.

6) In the do loop (210) the error test is employed to determine whether the integration step size needs to be doubled.

7) A linear interpolation process is used to obtain approximations of state variables at exact print interval point from those at the completed integration step. This process is carried out from do loop 13 to the label 33.

8) In the integrator coding KUTMER, the state variables of system equations are represented by the array variable Y(I). Similarly, the derivatives of the state variables are represented by the array variable DY(I). Moreover the array variables Y0(I),Y1(I),Y2(I) are the initial or previous values and estimates of state variables. The approximates of state variables at print time point is stored in YABC(I) array. Here *I* ranges from 1 to the maximum number of equations.

9)The values of the terms N, T, TEND, TAB and the initial values of the state variables are supplied by calling the HASP subroutine CONTRL. The purpose of calling subroutine AUX is to calculate the derivatives of state variables so that the estimates of state variables can be evaluated and finally determined. Subroutine OUT is used to store the simulation results. Both AUX and OUT are automatically written by the HASP Program Generator.

IMPLEMENTATION OF MODIFIED RUNGE-KUTTA-FEHLBERG METHOD FOR THE HASP SIMULATION PACKAGE

6.47 **Modified Runge-Kutta-Fehlberg Method.** According to the standard fifth order Runge-Kutta-Fehlberg method shown in paragraph 6.20, two modifications which are identical with those in the modified Runge-Kutta-Merson method were added. One is the error test for determining logical variables B1 and B2 which are used to control the estimation of state variables and to aid the option of integration step size. And another is the limitation of minimum integration step size h_{min} in the error test in order to avoid a possible infinite loop of the error evaluation.

6.48 **Error test of determining logical variables.** This test is added to determine the logical variables B1 and B2. The whole procedure is identical with that in KUTMER integrator, which has been discussed in paragraphs 6.43. Once the logical variables B1 and B2 are found and error estimate of state variables

$$ERROR \leqslant TOL$$

two logical variables B1 and B2 are used to determine whether the estimates of state variables are acceptable or not. If B1 or B2 is TRUE, these estimates are accepted and a new simulation time T is assigned by

$$T = T + H$$

then the simulation continues. If both B1 and B2 are FALSE, another error test for estimates of state variables still needs to be done in order to judge if the current integration step size will be doubled. This error test is given by:

$$\text{if } | ERROR | > \frac{TOL}{16}, \quad H = H$$

$$\text{if } 0 < | ERROR | < \frac{TOL}{16}, \quad H = 2H$$

6.49 **Control of error test by using minimum step size h_{min}.** A minimum integration step size for error test is assumed to be known in Runge-Kutta-Fehlberg algorithm. In the simulation process, once the integration step size is less than this value, the error test for the estimates of state variables will be passed and those estimates of state variables are considered to be acceptable. The procedure about this is given by:

$$\text{if } | ERROR | \geqslant TOL$$

the integration step is halved:

$$h = \frac{h}{2}$$

if $h < h_{min}$ jump out from error test of estimates

if $h \geqslant h_{min}$ carry on error test

6.50 **Structure of KUTFEH integrator.** This modified Runge-Kutta-Fehlberg method is implemented in the HASP simulation package and called as KUTFEH integrator. The main structure of KUTFEH integrator is identical with that of KUTMER integrator, which is shown in figure 6.26 and the flow chart of KUTFEH integrator is shown in figure 6.27.

The listing of KUTFEH integration subroutine is shown in Appendix D of this thesis.

SIMULATION EXAMPLE OF A HYDRAULIC SYSTEM USING SINGLE STEP INTEGRATORS

6.51 **Introduction.** Three single step method numerical integrators including RK4, KUTMER and KUTFEH have been implemented in the HASP simulation package. They may simulate the dynamic behaviour of both continuous and discontinuous systems which are modelled by cubic smoothing technique or conditional statement technique (IF STATEMENTS method). In order to prove this, one case study is presented below, in which different integrators and different component models are used to simulate dynamics of a linear actuator system.

6.52 In order to simulate the dynamics of a linear actuator system, a linear actuator model AL00 and two frictionless pipe models PI05 and PI06 have been developed in the HASP simulation package. Since the discontinuities of these models are smoothed by employing cubic smoothing technique, these models become continuous functions in whole simulation process, but the coding of components modelled by cubic smoothing technique is normally difficult to read and to model for the users (even for writers). Fortunately, three integrators developed above are also suitable for the simulation of discontinuous models using IF STATEMENTS method which is easy to understand and to write. Here discontinuous models ALR0, PIR5 and PIR6 are used to replace above continuous models AL00, PI05 and PI06 respectively and these three component models with IF STATEMENTS are given below.

6.53 **Linear Actuator Model.** Consider a linear actuator which is shown in figure 6.28. This actuator may be inclined at any angle between plus and minus 90 degrees. Special modelling methods have been developed for stick/slip friction and the representation of the non-linear or discontinuous functions. The dynamic model of the actuator includes a second order differential equation and therefore there are two state variables in it, one for actuator displacement and the other for velocity. The model assumptions are that the mass of the moving parts of the actuator and any load attached to the actuator may be lumped together as a single term; the stiction and coulomb friction forces are constant and their values only depend on the velocity of the actuator. The representation about friction forces is shown in figure 6.29.

6.54 **Equations of motion.** The differential equation described the motion of the actuator is given by:

$$M\frac{dv}{dt} + f_v\, v + k\, x + F_f + m\, g\, \sin\theta = P_{in}\, A_1 - P_{out}\, A_2 \qquad \ldots\ldots (6.37)$$

where $\frac{dv}{dt} = \frac{d^2x}{dt^2}$

and F_f denotes the friction force and here coulomb and stiction terms are incorporated:

$$F_f = \begin{cases} -sign\,(F_{ap})xF_s & \text{if } v = 0 \\ -sign\,(v)xF_c & \text{if } |v| > 0 \end{cases} \qquad \ldots\ldots (6.38)$$

where the net applied force Fap is given by:

$$F_{ap} = P_{in}\, A_1 - P_{out}\, A_2 - m\, g\, \sin\theta \qquad \ldots\ldots (6.39)$$

The second order differential equation is rewritten as the following two first order differential equations

$$\frac{dx}{dt} = v$$

defining the time derivative of the piston displacement and

$$\frac{dv}{dt} = \frac{1}{M}(P_{in}\, A_1 - P_{out}\, A_2 - f_v\, v - m\, g\, \sin\theta - k\, x - F_f) \qquad \ldots\ldots (6.40)$$

defining the time derivative of the piston velocity and the non-linear friction force

F_f is expressed in equation (6.37).

6.55 **Modelling of end stops.** The equation of motion for the actuator piston between the end stops ($0 < X < X_{max}$) is given by the equation (6.37). However at the end stops, the equation of motion is not suitable for the actuator because the real end-stop reaction forces acting on the actuator piston are not considered. If there is no any modification for the equation of motion, in the low end-stop the displacement of the actuator might be less than zero and in the high end-stop might be greater than X_{max}. That is not true in the practical situation. A better method for coping with this kind of the difficulty is to use the "IF STATEMENTS" technique in the component model subroutine so that the displacement "overshoot" or "undershoot" existed in the simulation can be limited, and the velocity of the piston also can be limited to be zero when the actuator hits a stop. The modelling for end-stops can be given by:

$$\text{if } X > X_{max} \text{ or } X < 0 \quad \dot{v} = 0 \qquad \ldots\ldots (6.41)$$

$$\text{if } X < 0 \qquad \text{then } X = 0 \qquad \text{and } v = 0 \qquad \ldots\ldots (6.42)$$

$$\text{if } X > X_{max} \qquad \text{then } X = X_{max} \text{ and } v = 0 \qquad \ldots\ldots (6.43)$$

6.56 **Computer coding considerations for end stops.** There are two state variables in the model subroutine of an linear actuator with constant load, one for the displacement x and another for the velocity v. The computer modelling of end stops is given by:

```
C ***** PHYSICAL CONSTRAINED CONDITIONS
C
      IF(X.GT.CON(7)) THEN
       X=CON(7)
       V=0.D0
       VDOT=0.D0
       GOTO 111
```

```
      END IF
      IF(X.LT.0.D0) THEN
       X=0.D0
       V=0.D0
       VDOT=0.D0
       GOTO 111
      END IF
  111 CONTINUE
C
```

Where CON(7) is the actuator stroke. The coding of this actuator model is shown in a model subroutine ALR0.FOR in the HASP model library and its documentation is detailed in Appendix C of this thesis.

6.57 A Frictionless Pipe Model with Air Release and Variable Pipe Volume.

This model considers the dynamic transient of the pressure in a pipe for which the friction effects are neglected. The model also includes a representation of air release when the pipe is operating at low pressure. This effect will be a reasonably accurate representation of cavitation in many cases. Since the pipes described by this model may be connected to an actuator where the internal volume of fluid can not be neglected, the effects of variable volume of fluid on the transient pressure in a pipe have to be considered. The change rate of the pressure in a pipe depends on the flow rates supplied from other adjacent models and it can be expressed as follows:

$$\frac{dP}{dt} = \frac{\beta_e}{V_p + V_{IN}} \sum Q_i \qquad \ldots\ldots (6.44)$$

According to the mechanism of the air release proposed by Dugdale[18], the representation of the effective bulk modulus of pipe can be given by:

1) if the operating pressure P is larger than the saturated pressure P_{sat}, i.e. $P > P_{sat}$ the fluid effective bulk modulus is:

$$\beta_e = \beta_f$$

2) if $-1 < P \leqslant P_{sat}$, β_e can be expressed as:

$$\beta_e = \frac{1}{\frac{1}{\beta_f} + \frac{d_0 (P_{sat} - P)}{n (P+1)^2}} \qquad \ldots\ldots (6.45)$$

where the expression of β_e is a discontinuous non-linear function and a minimum value for operating pressure must be limited as:

$$P = -0.99 \text{ bar}, \quad \text{if } P < -1 \text{ bar}$$

6.58 **Computer coding considerations.** Above discontinuous functions are coded by IF STATEMENTS method. When the operating pressure is below saturation pressure, the effective bulk modulus is performed by following coding:

```
C ***** P IS BELOW SATURATION PRESSURE
C
      BAIR=CON(4)*(P+1.D0)**2/(CON(6)*(CON(5)-P))
      BEFF=1.D0/(1.D0/CON(1)+1/BAIR)
```

where the constant CON(1) is the fluid bulk modulus and CON(4) is the polytropic index. The constant CON(5) is the pressure at which oil is saturated with air and CON(6) is the proportion of air by volume dissolved in oil at saturation pressure.

When the operating pressure is above saturation pressure, the effective bulk modulus is performed by following coding:

```
C
C ***** P IS ABOVE SATURATION PRESSURE.
```

```
C
      BEFF=CON(3)
```

where the CON(3) is a constant of the effective bulk modulus of fluid divided by pipe volume. The physical constrained condition of the pressure in a pipe is:

```
      IF(P.LE.-1.D0)P=-9.99D-1
```

A full description of this model (PIR6) is shown in Appendix C of this thesis.

6.59 **A Frictionless Pipe Model with Air Release.** Assume the internal volume of components connected to a pipe will not affect pressure change in the pipe. In this case, the dynamics of a pipe is expressed by:

$$\frac{dP}{dt} = \frac{\beta_e}{V_p} \Sigma Q_i \qquad \ldots\ldots (6.46)$$

The effective bulk modulus β_e of fluid and its coding are identical with that in model PIR6. For this reason, this model (PIR5) can be obtained by setting $VIN = 0$ in the model PIR6.

6.60 **Circuit Description.** Figure 6.16 shows a linear actuator system being investigated. The circuit data details of performing the program generation using the HASP program generator are obtained from the computer model linking diagram of the system shown in figure 6.30 and they are given below.

```
10
PM0001    12
PU0001    8  9  12
PIR501    2  9  10
PC0101    10 11
TK0301    8  11 5
DC1T01    1  2  3  4  5
PIR601    6  3  1
ALR001    6  7  13 1  2
PIR602    7  4  2
DE0101    1
```

The prime mover of this linear actuator system supplies a constant speed to the pump as a flow variable on link 12. The main pump delivery flow and pressure are respectively the flow and effort variables on link 9. The system pressure is the effort variable on links 9, 10 and 2. The flow rate reliefed from a relief valve is available as the flow variable on links 10 and 11. The net flow rate through the directional control valve is supplied as the flow variable on link 2. The operating position of the directional control valve is available on link 1. The flow rate and pressure to the piston and rod sides of the actuator are respectively supplied as the flow and effort variables on links 6 and 7. Volume changes for both sides of the actuator are transmitted to the pipe models connected to the actuator and are available on signal links 1 and 2. Here PIR6 model is used twice in the circuit.

6.61 **Test Simulation and Results.** In order to test the performance of the end stop model using the IF STATEMENTS method, the actuator is extended to fully stroke position and it is necessary to remain on an end stop for several seconds. And then the actuator is retracted to part stroke position. The operating position of the directional control valve is shown in figure 6.31. The actuator displacement is shown in figure 6.32 and the pressure responses in actuator piston side and rod side are shown in figures 6.33 and 6.34 respectively. The simulation data used is given in tables 6.1 to 6.2.

6.62 **A comparison of simulation speeds for three single step method integrators.** The mathematical model of this linear actuator circuit is a typical stiff and discontinuous system. Since fourth order Runge-Kutta algorithm is a fixed step method, it takes long long time for the simulation of this system using RK4 integrator. However, Runge-Kutta-Merson and Runge-Kutta-Fehlberg algorithms are variable step methods, and hence simulations of this circuit is very fast. The simulation results obtained by these integrators are identical and a comparison of simulation speeds for them is shown in the following table:

Table 6.A: Comparison of simulation speed

Working Condition	Algorithm	Step Length (seconds)	CPU time (seconds)
Ttotal = 12 sec.	RK4	$5.x10^{-6}$	86468
Tprint = 0.02 sec.	KUTMER		407.6
	KUTFEH		398.5

6.63 **A comparison between execution times for the two types of end stop logic.** Above three numerical integrators also can be suitable for the simulation of continuous systems which consist of current HASP component models. For instance, using AL00 model whose end stops logic is modelled by employing the cubic smoothing technique replaces the model ALR0 whose end stop logic is modelled by IF STATEMENTS. And the pipe model PIR5 and PIR6 are also replaced by the PI05 and PI06 respectively, a relevant computer linking block diagram for this simulation is shown in figure 6.17. The simulation results obtained are identical with those shown in figures 6.32 to 6.34. However, the execution times of system simulation are different due to different end stop logic used. A comparison between execution times for two types of end stop logic is shown in the table 6.B below.

Table 6.B: Comparison of execution time for different end stop logic

Actuator Model end stop logics	Algorithm	Step Length (seconds)	CPU time (seconds)
Ttotal = 12 sec. Tprint = 0.02 sec.			
Cubic smoothing technique	GEAR	$h_{max}=10^{30}$ $h_{min}=10^{-6}$	1212
	KUTMER	$h_{max}=10^{30}$ $h_{min}=10^{-6}$	713
	KUTFEH	$h_{max}=10^{30}$ $h_{min}=10^{-6}$	670.3
IF STATEMENTS method	KUTMER	$h_{max}=10^{30}$ $h_{min}=10^{-6}$	407.6
	KUTFEH	$h_{max}=10^{30}$ $h_{min}=10^{-6}$	398.5

Parameter	Sign	Value	Unit
Prime Mover	ω_p	1.5×10^3	rev/min
maximum pump displacement	X_p	$5. \times 10^{-2}$	l/rev
fraction of full displacement	k_f	1	
relief valve cracking pressure	P_c	1.1×10^2	bar
relief valve constant	k	0.5	(L/s)/bar
directional control valve			
orifice Constant	C_1	1.4556×10^{-1}	
	C_2	1.4809×10^{-1}	
	C_{v1}	1.3399×10^{-1}	
	C_{v2}	1.3596×10^{-1}	
rated pressure	P_{rate}	$1. \times 10^2$	bar
Pipe (PIR5)			
bulk modulars	β	7×10^3	bar
pipe diameter	d	1.5×10^1	mm
pipe length	L	5.88	m
initial pressure	P_0	1.6×10^1	bar
saturation press.	P_{sat}	0	bar
proportion of dissolved air	d_0	0.1	

Table 6.1 Parameters for simulation test of a hydraulic system using single step integration methods

Parameter	Sign	Value	Unit
Pipe (PIR6) No.1			
bulk modulars	β	$7x10^3$	bar
pipe diameter	d	$1.5x10^1$	mm
pipe length	L_1	5.88	m
initial pressure	P_0	$1.6x10^1$	bar
saturation press.	P_{sat}	0	bar
proportion of dissolved air	d_0	0.1	
Pipe (PIR6) No.2			
bulk modulars	β	$7x10^3$	bar
pipe diameter	d	$1.5x10^1$	mm
pipe length	L_2	$1.247x10^1$	m
initial pressure	P_0	$2.1x10^1$	bar
Actuator			
piston diameter	D	5.08	cm
rod diameter	d	2.5	cm
mass	M	$8.7x10^2$	kg
stiction force	F_s	$2.x10^2$	N
colomb fric. force	F_c	$1.x10^2$	N
viscous coeff.	f_v	$5.x10^3$	$N/(m/s)$
piston stroke	X_{max}	0.61	m
initial displac.	X_0	0.0	m
initial speed	v_0	0.0	m/v
tank pressure	P_t	0.0	bar

Table 6.2 Parameters for simulation test of a hydraulic system using single step integration methods

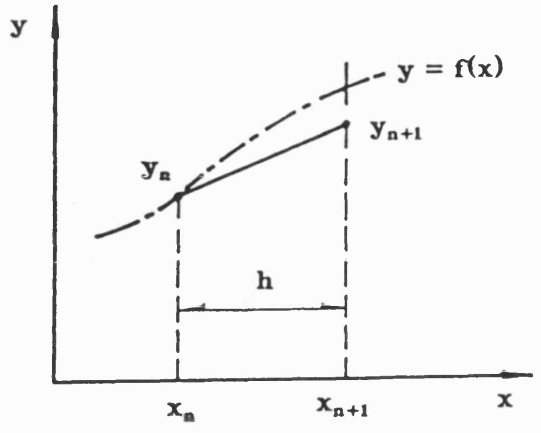


Figure 6.1 Simple Euler method

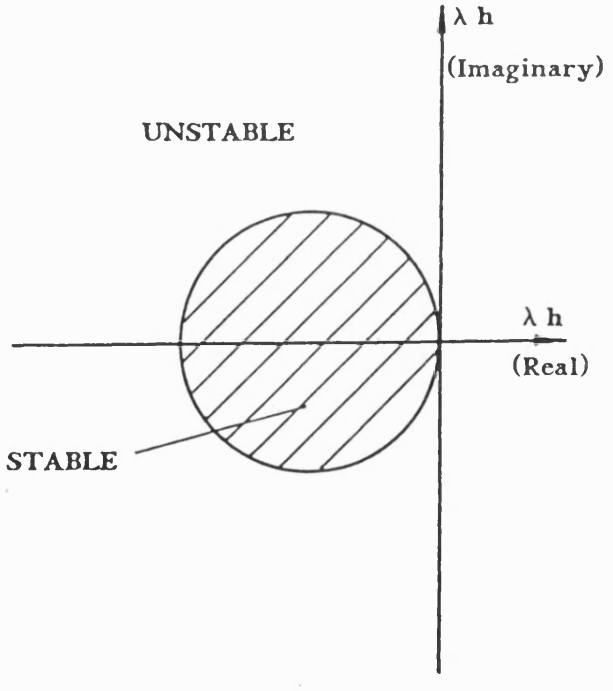


Figure 6.2 Stability region of simple Euler method

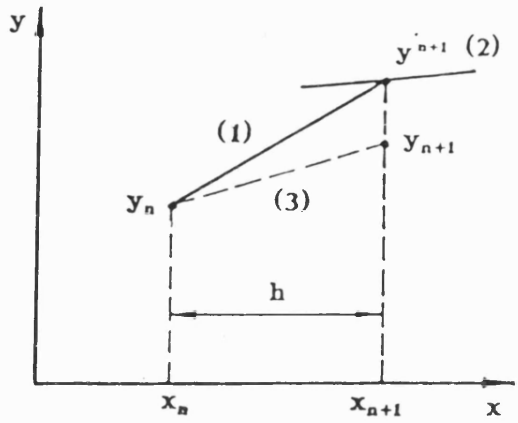
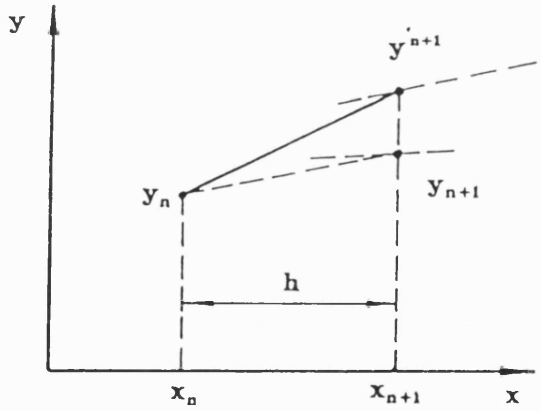


Figure 6.3 Improved Euler method



Figrue 6.4 Backward Euler method

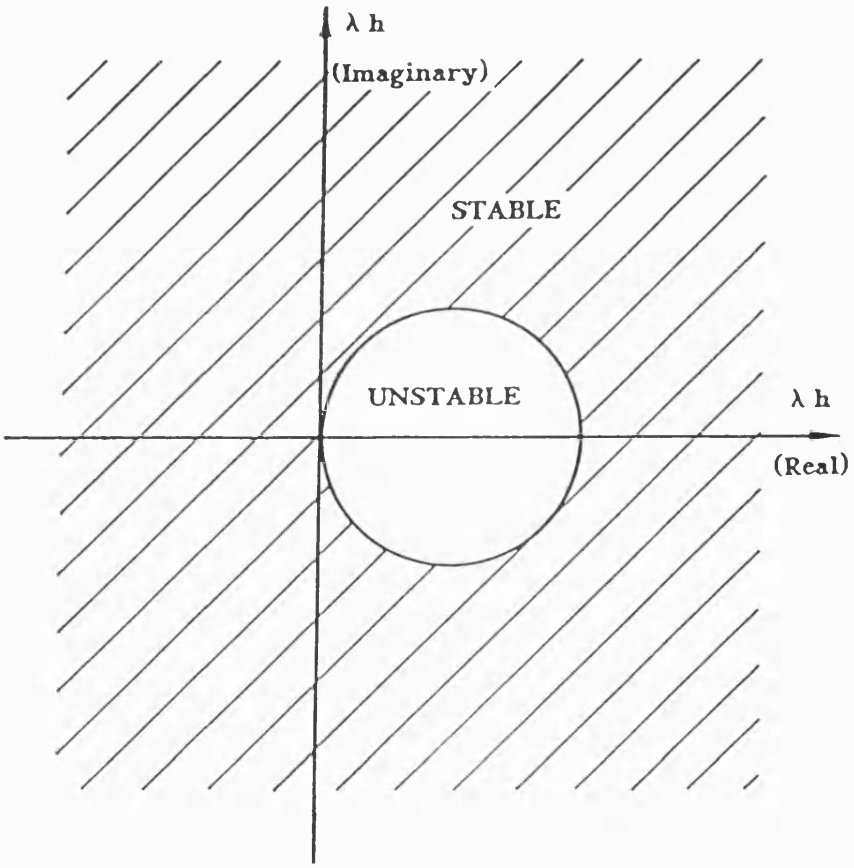


Figure 6.5 Stability region of backward Euler method

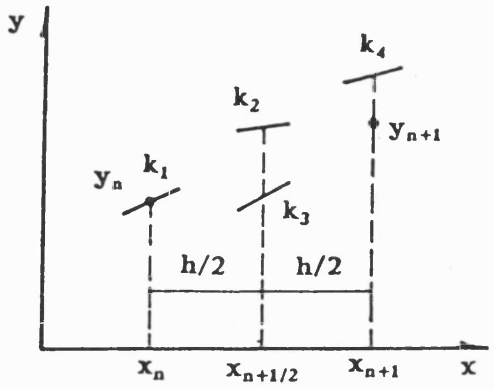


Figure 6.6 Fourth order Runge-Kutta method

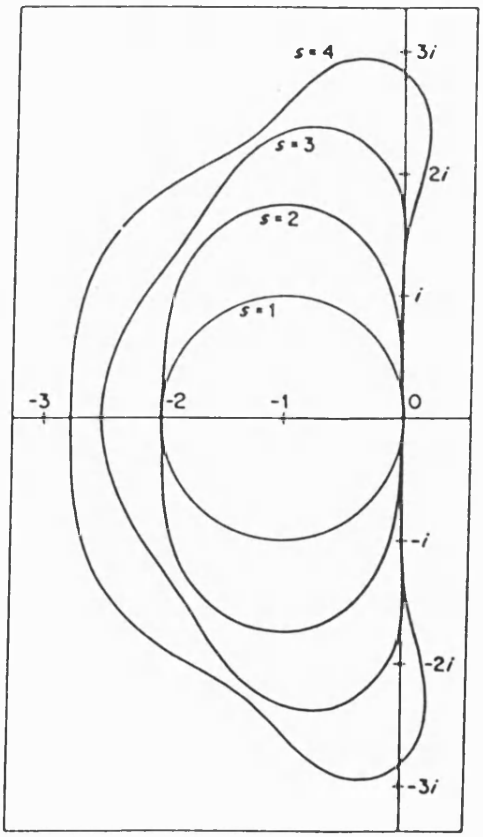


Figure 6.7 Stability diagram for Runge-Kutta integration methods up to order four

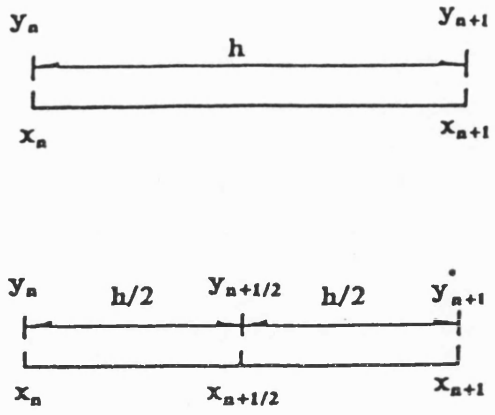


Figure 6.8 Representation for the estimation of y_{n+1} and y^*_{n+1}

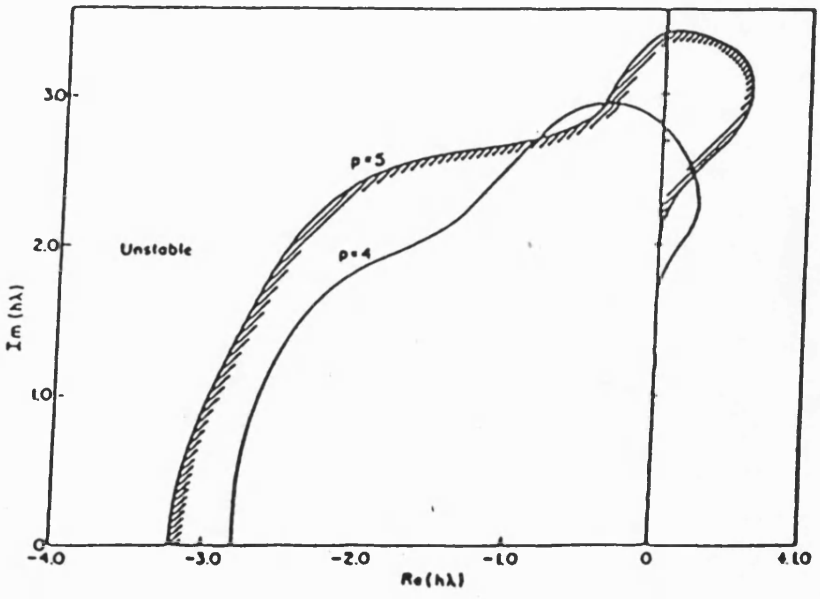


Figure 6.9 Stability region for fifth order Runge-Kutta method

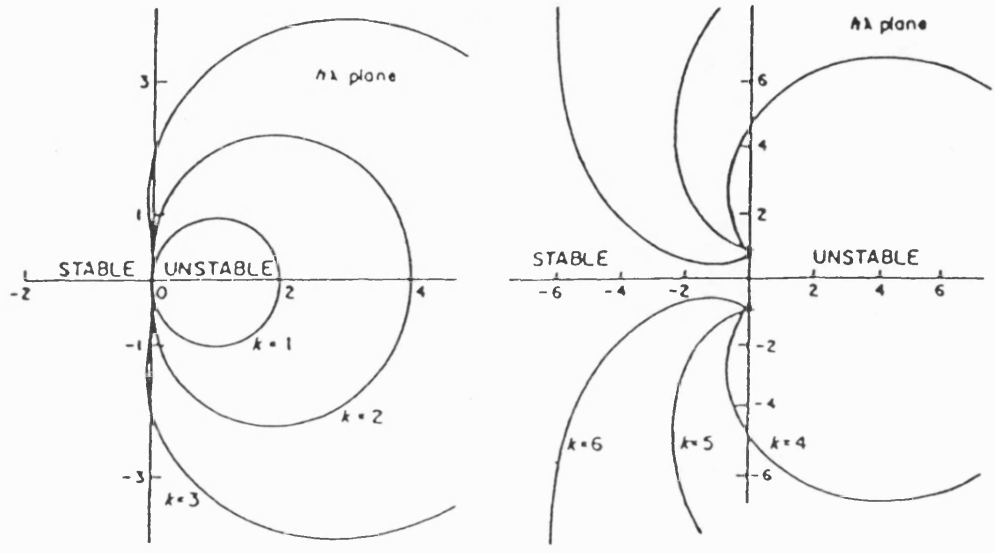


Figure 6.10 Stability region for Gear's method

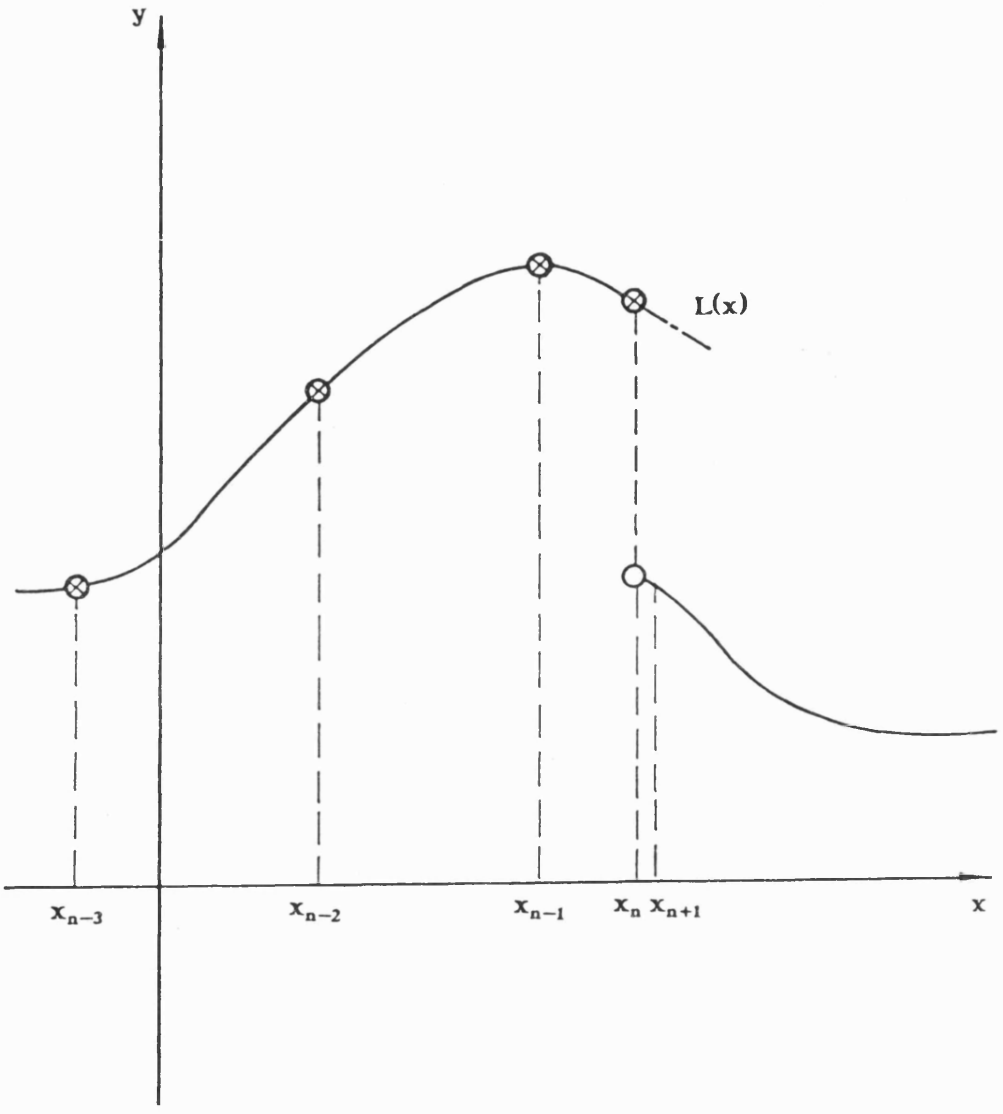


Figure 6.11 Representation of a discontinuous function

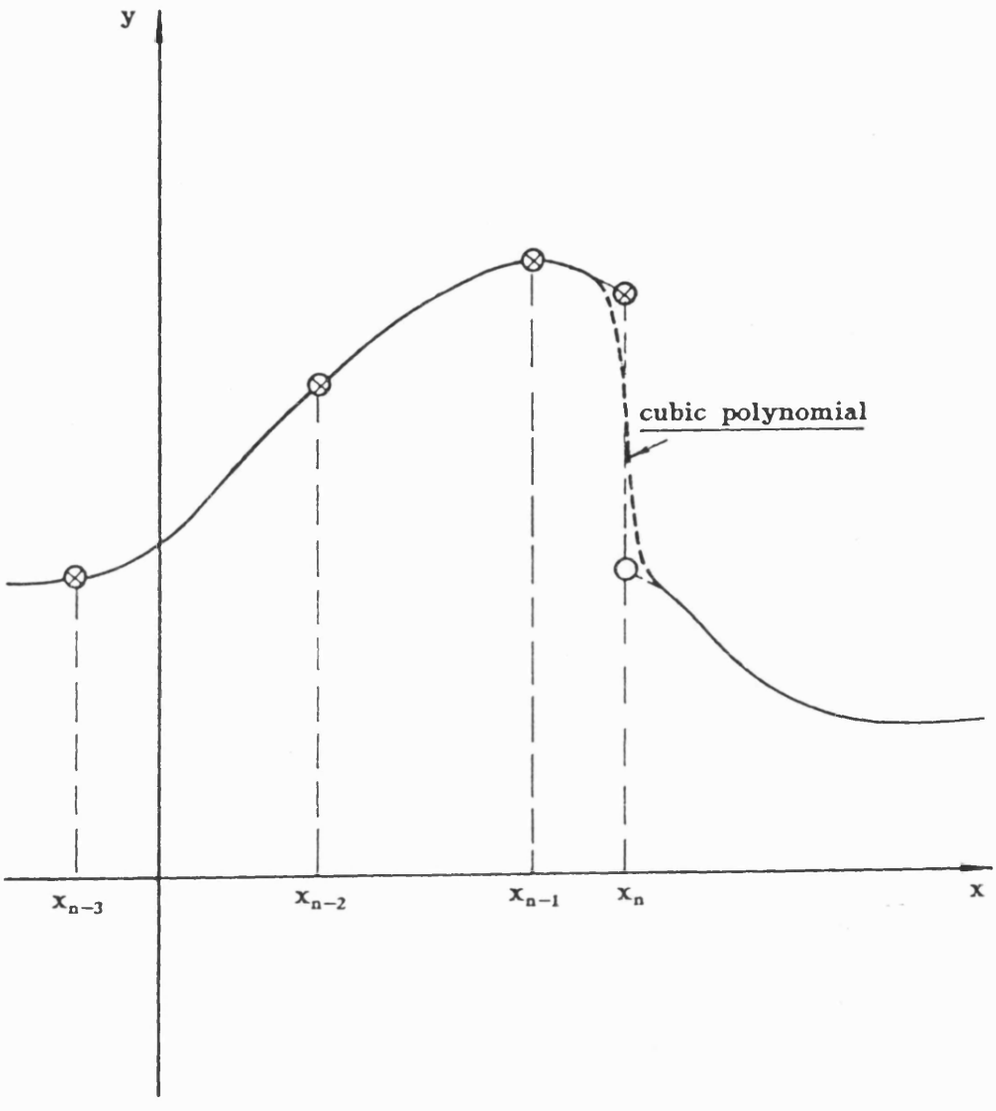


Figure 6.12 Representation of solving discontinuous problems by cubic smoothing technique

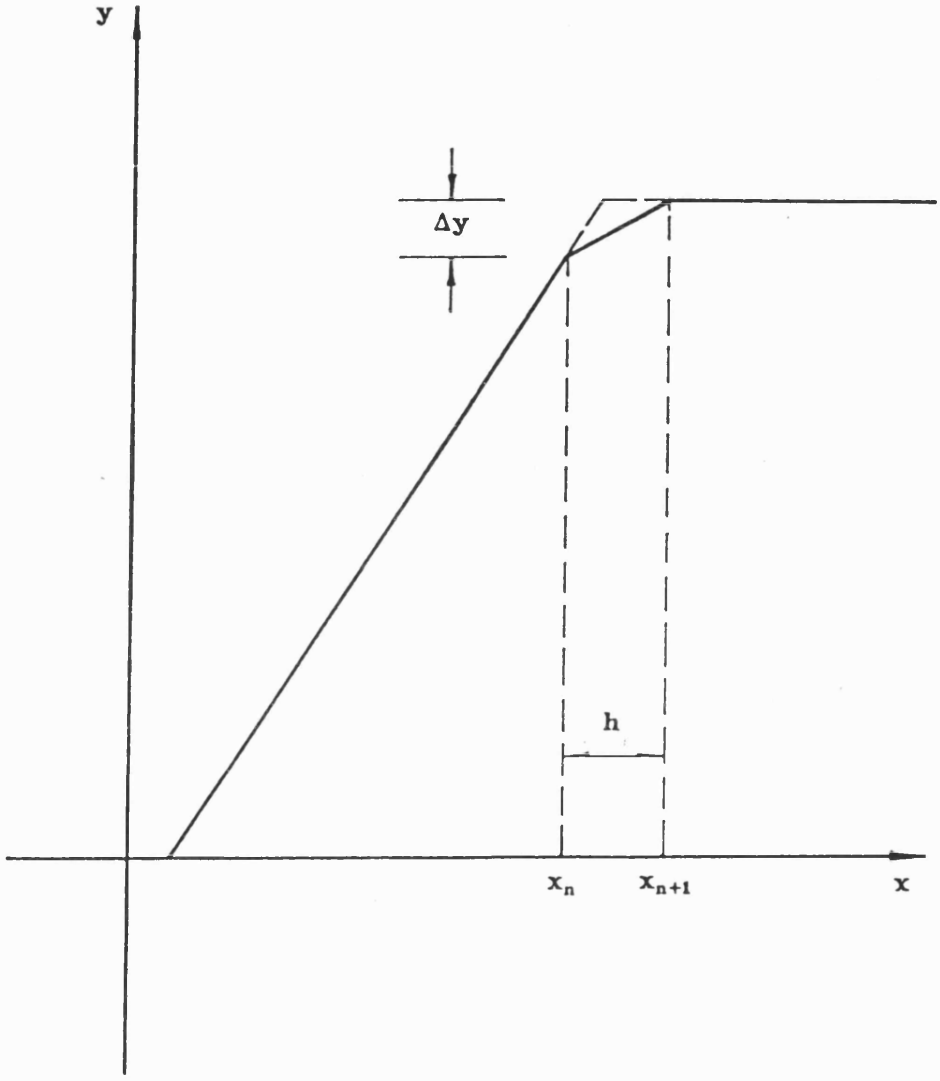


Figure 6.13 Error of solution on a straight line between two time points

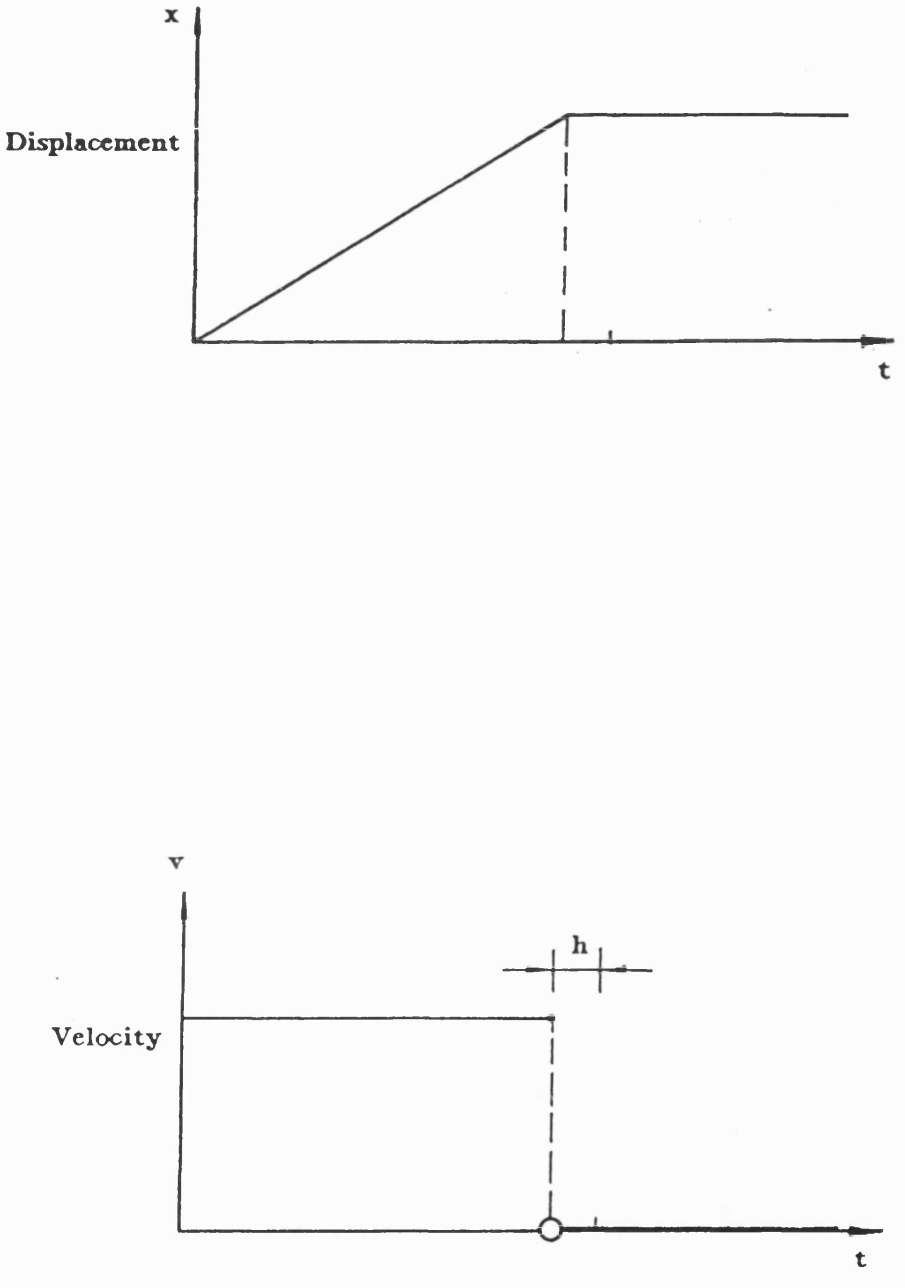


Figure 6.14 Representation of one kind of discontinuity

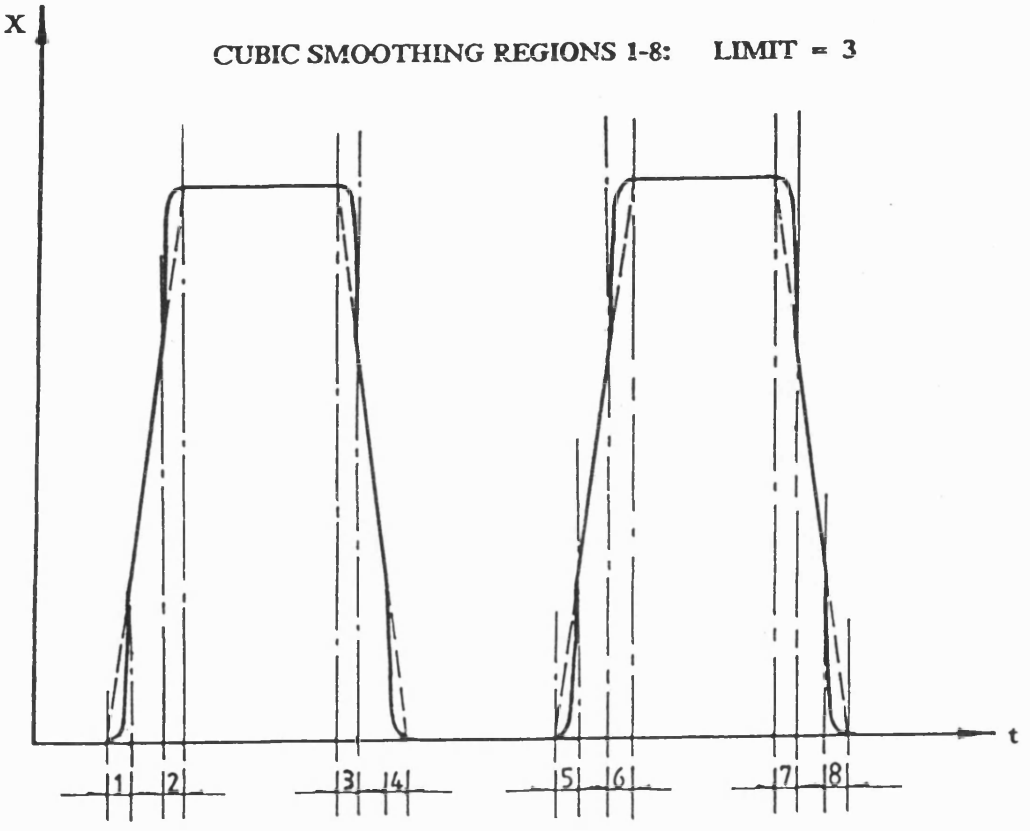


Figure 6.15 Representation for discontinuous region in a duty cycle model

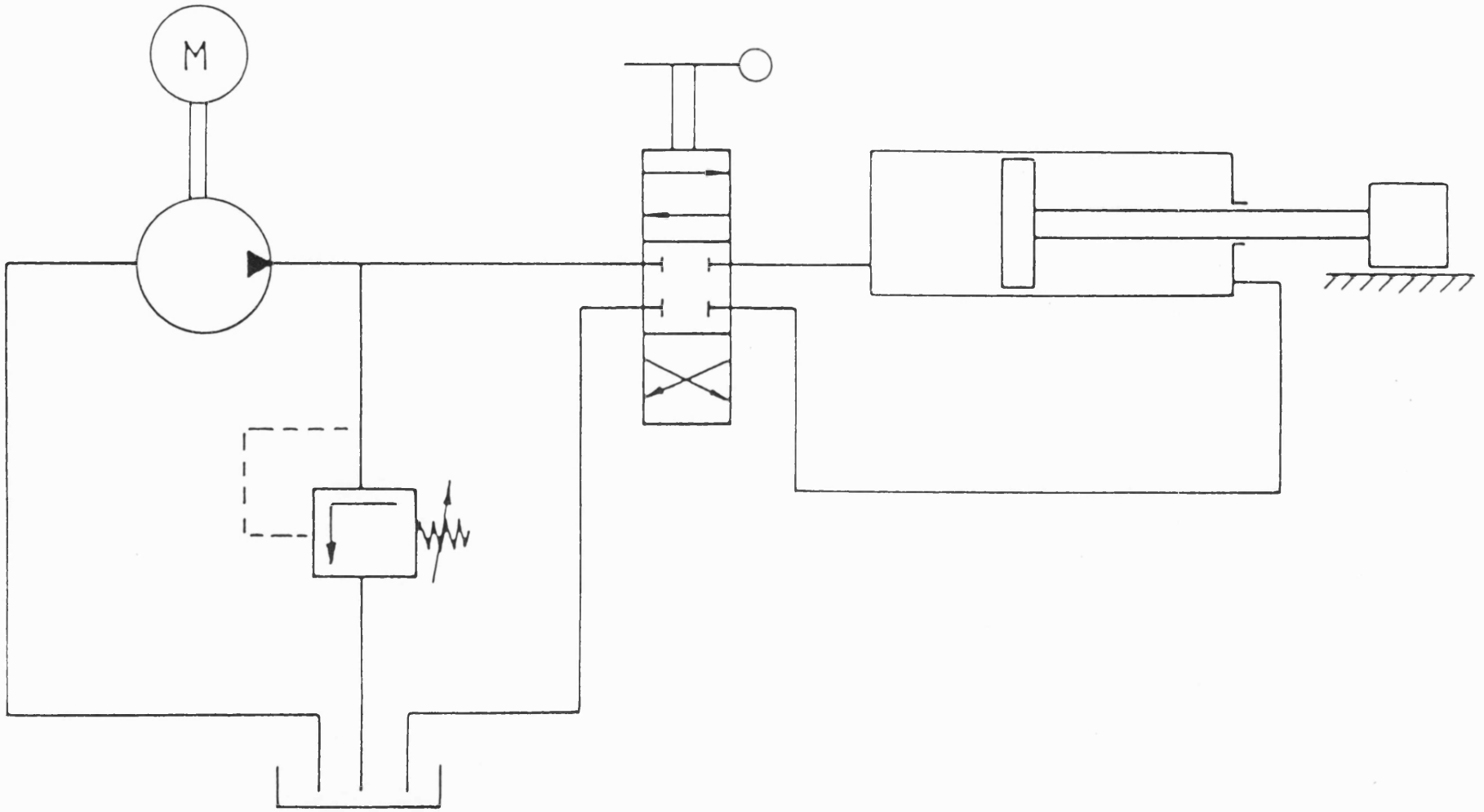


Figure 6.16 A linear actuator circuit

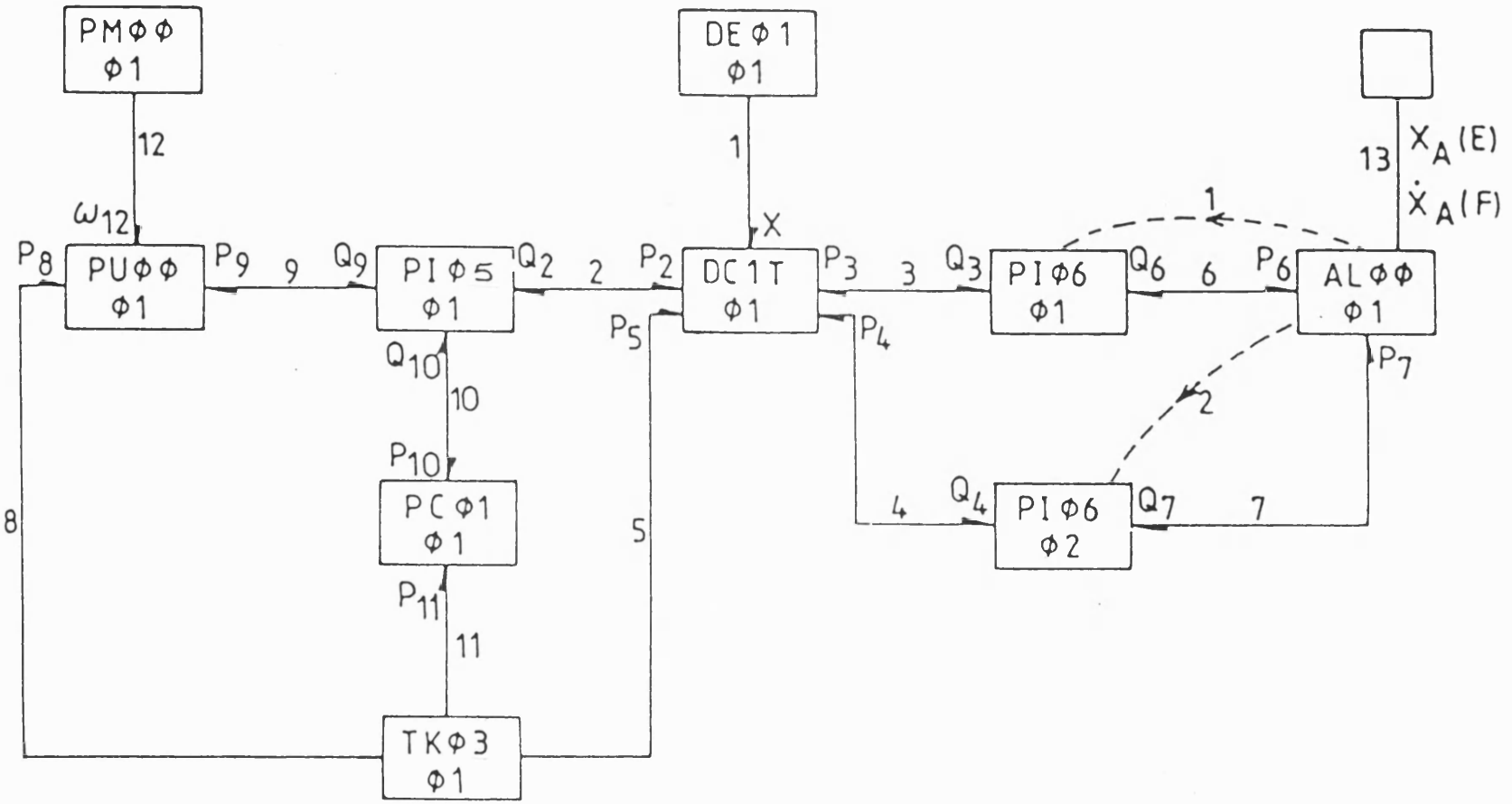
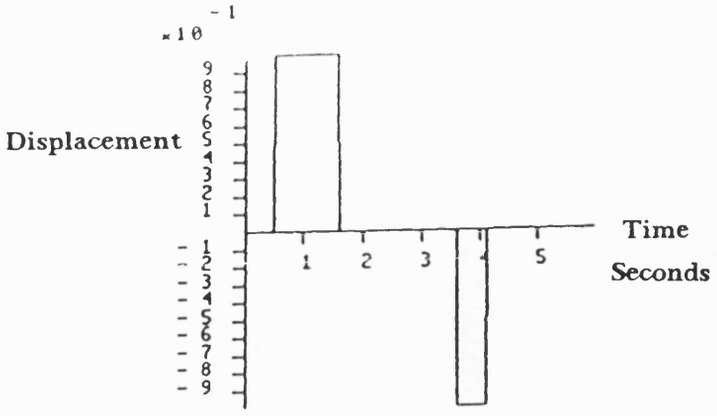
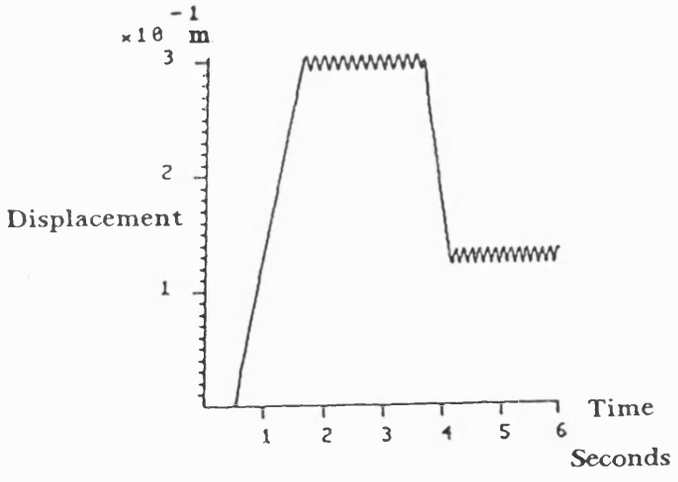


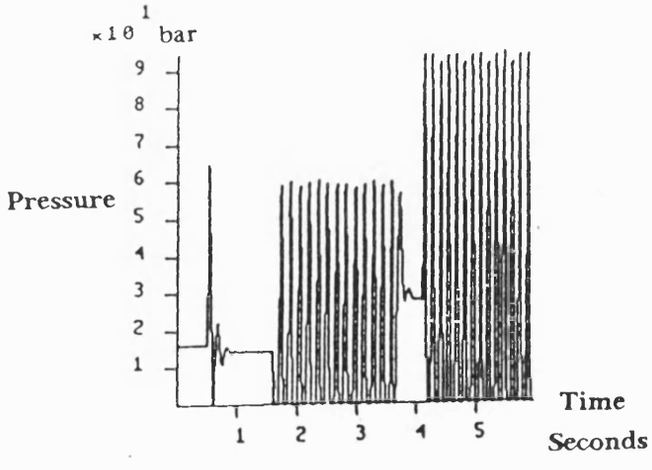
Figure 6.17 Computer linking diagram for linear actuator circuit simulation



(a) Manual position

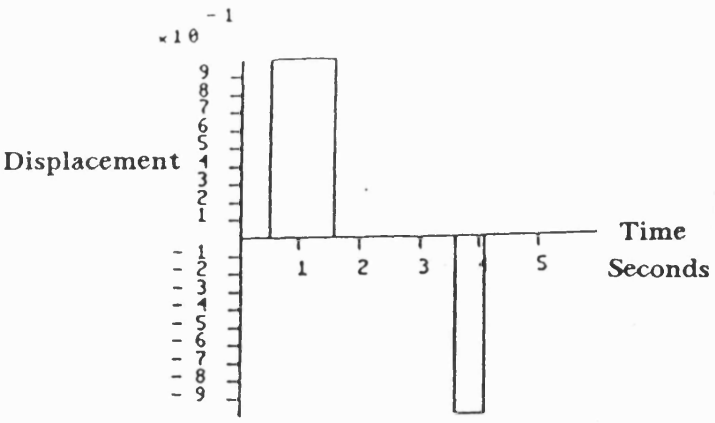


(b) Actuator displacement

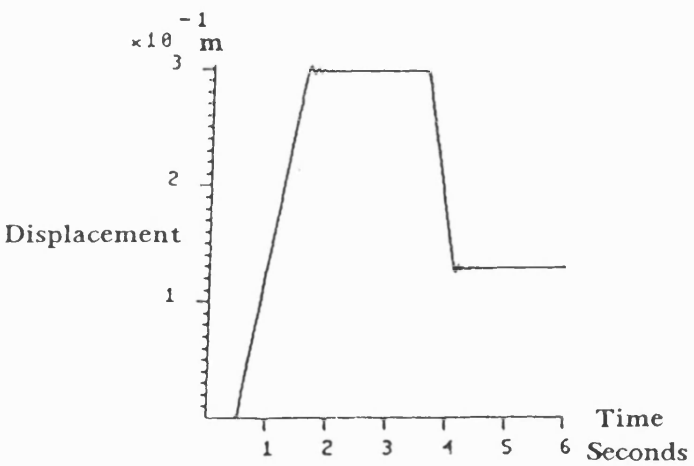


(c) Piston side pressure

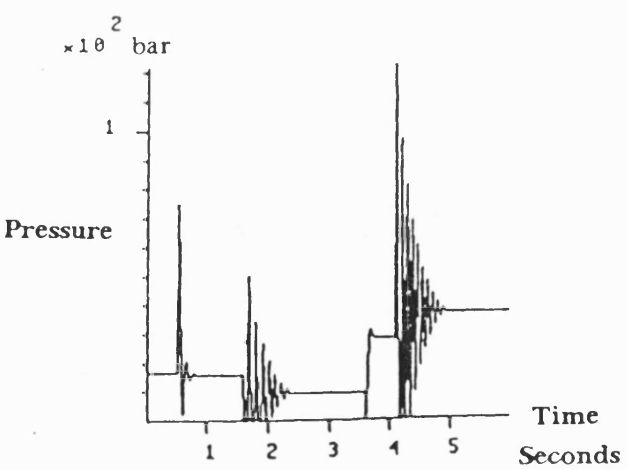
Figure 6.18 Simulation results of a linear actuator using model DE01



(a) Manual position

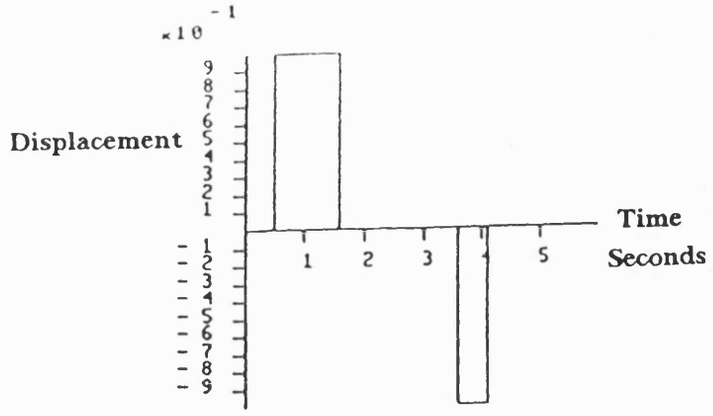


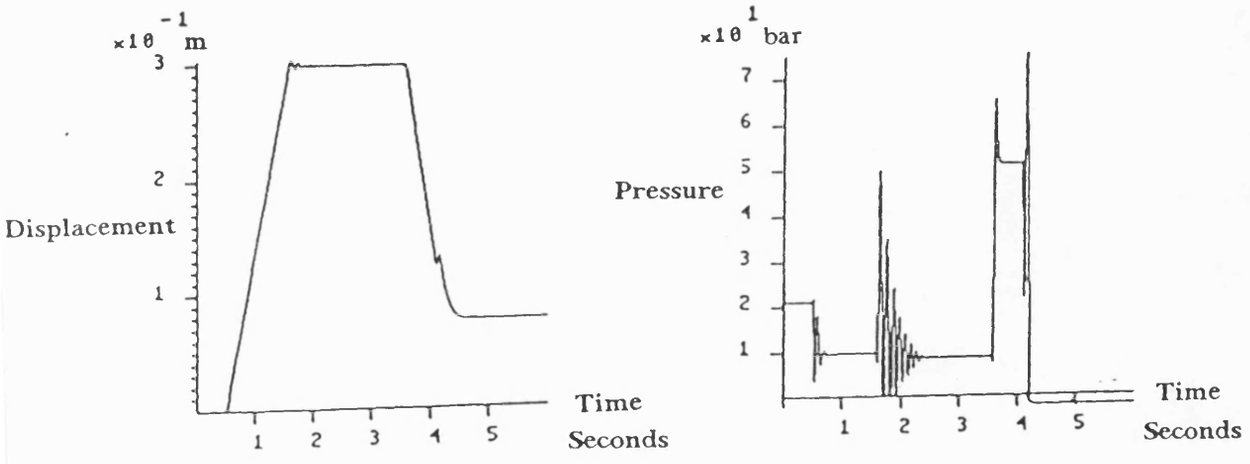
(b) Actuator displacement



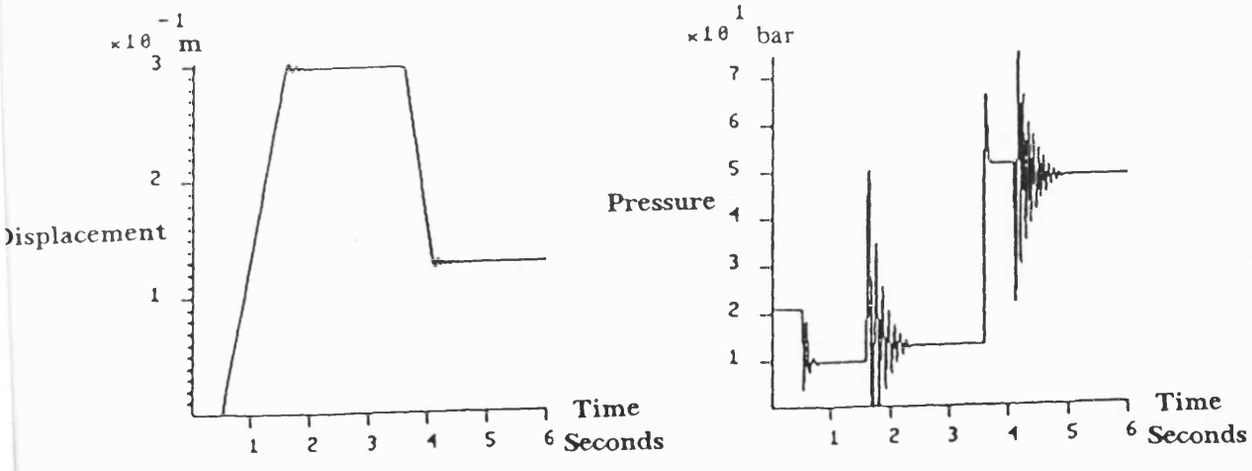
(c) Piston side pressure

Figure 6.19 Simulation results of a linear actuator using model DER1





(a) $H_{min} = 10^{-3}$



(b) $H_{min} \leq 10^{-4}$

Figure 6.20 Simulation results for the choice of h_{min}

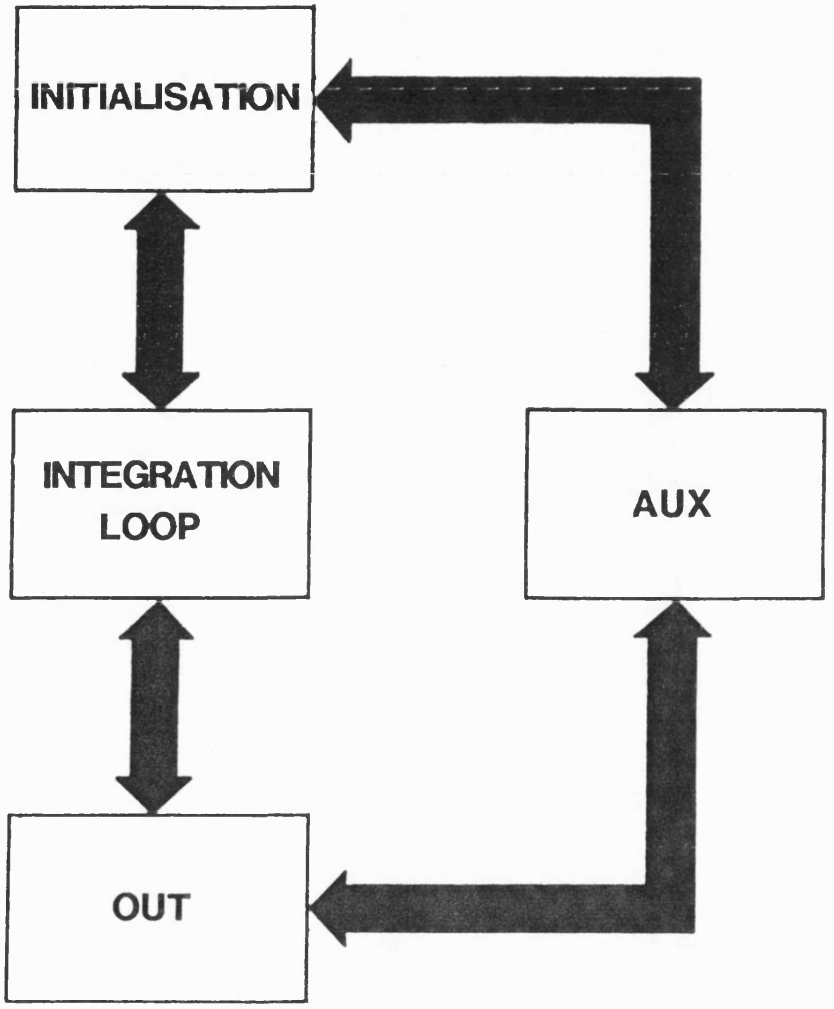


Figure 6.21 Structure of RK4 integrator

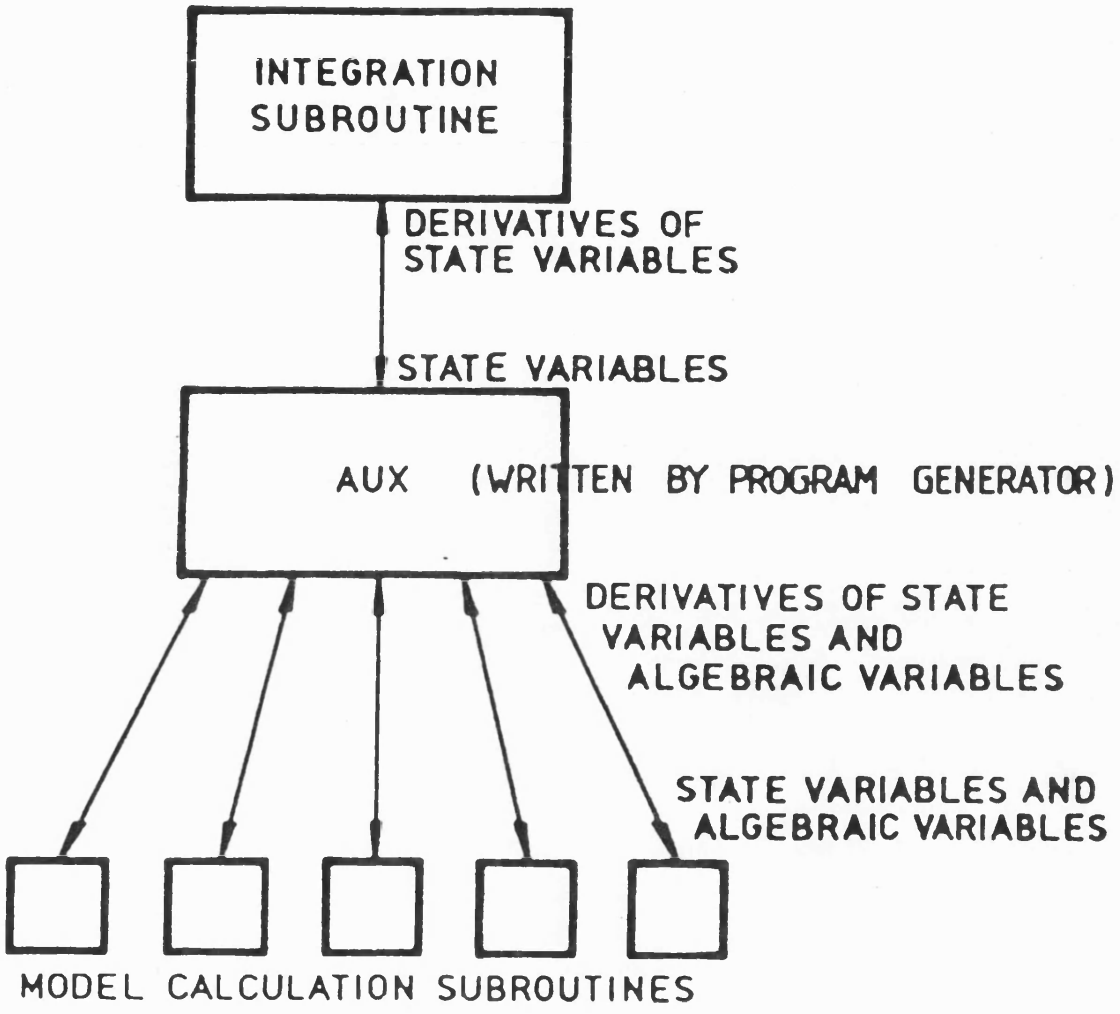


Figure 6.22 Interaction between the integration subroutine, AUX subroutine and the component model subroutines

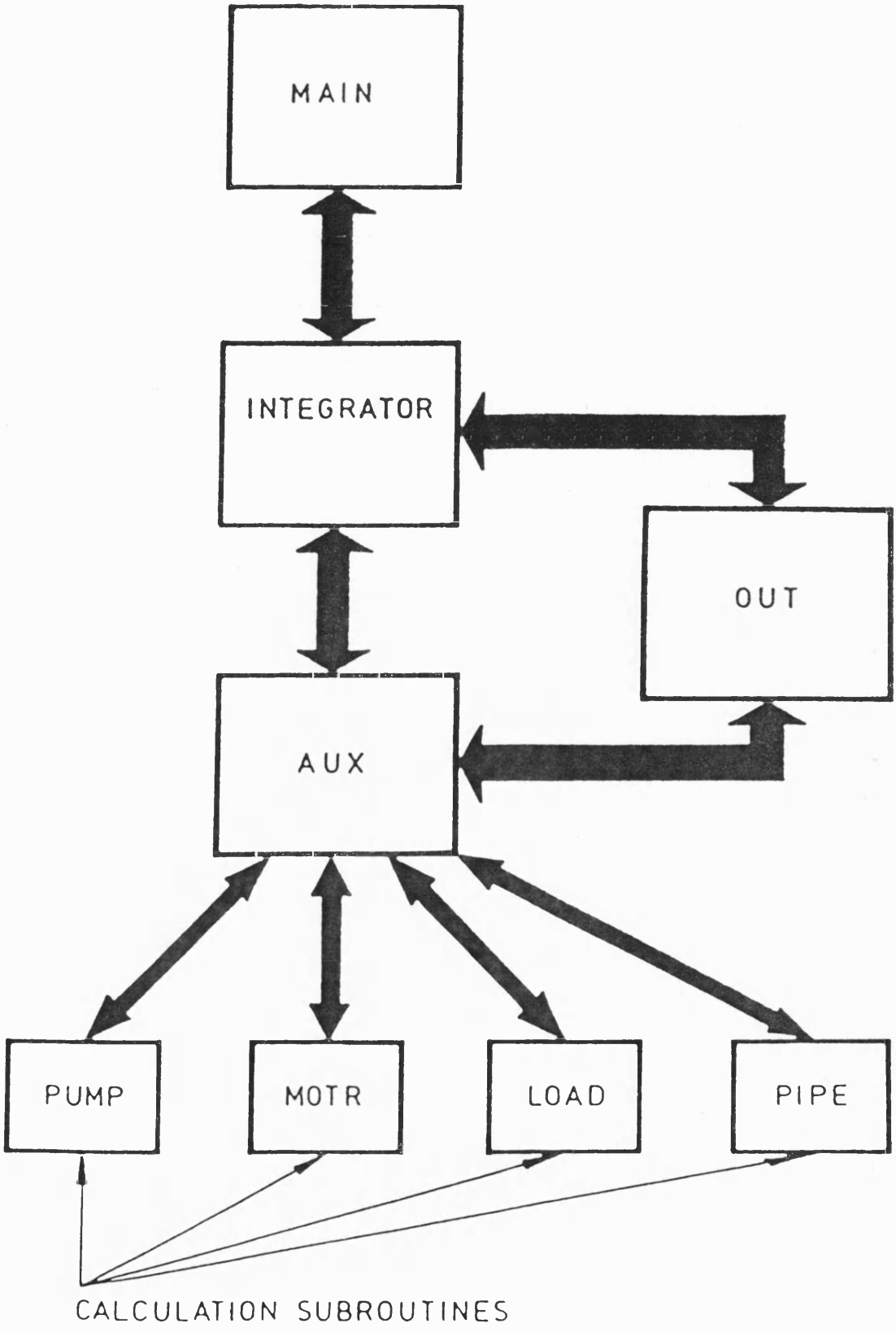


Figure 6.23 The integration process

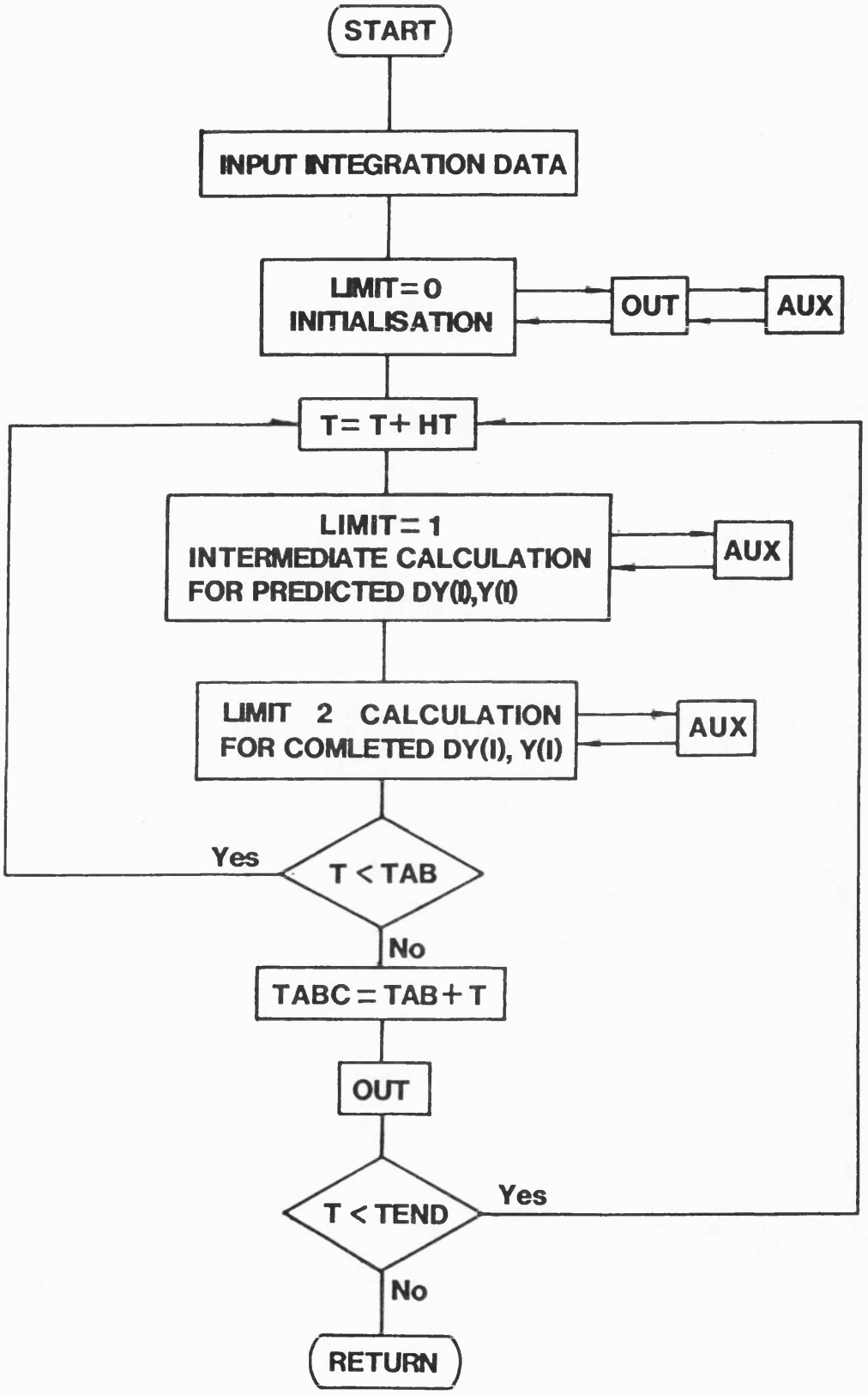
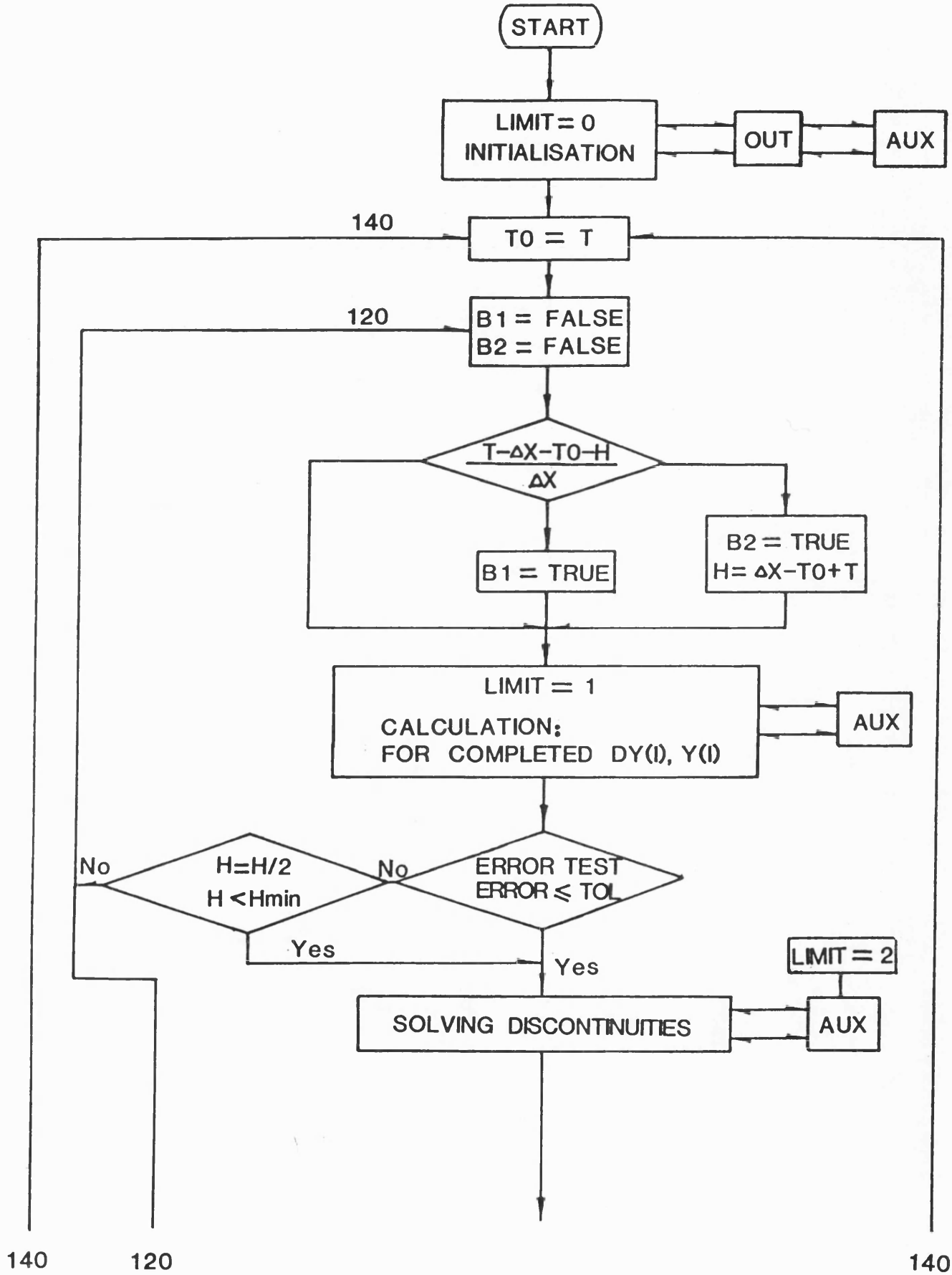


Figure 6.24 Flow chart of RK4 integrator coding


START
LIMIT = 0
INITIALISATION
OUT
AUX
140
T0 = T
120
B1 = FALSE
B2 = FALSE
T-ΔX-T0-H
ΔX
B1 = TRUE
B2 = TRUE
H= ΔX-T0+T
LIMIT = 1
CALCULATION:
FOR COMPLETED DY(I), Y(I)
AUX
No
H=H/2
H <Hmin
No
ERROR TEST
ERROR ≤ TOL
Yes
Yes
LIMIT = 2
SOLVING DISCONTINUITIES
AUX
140
120
140

Continue

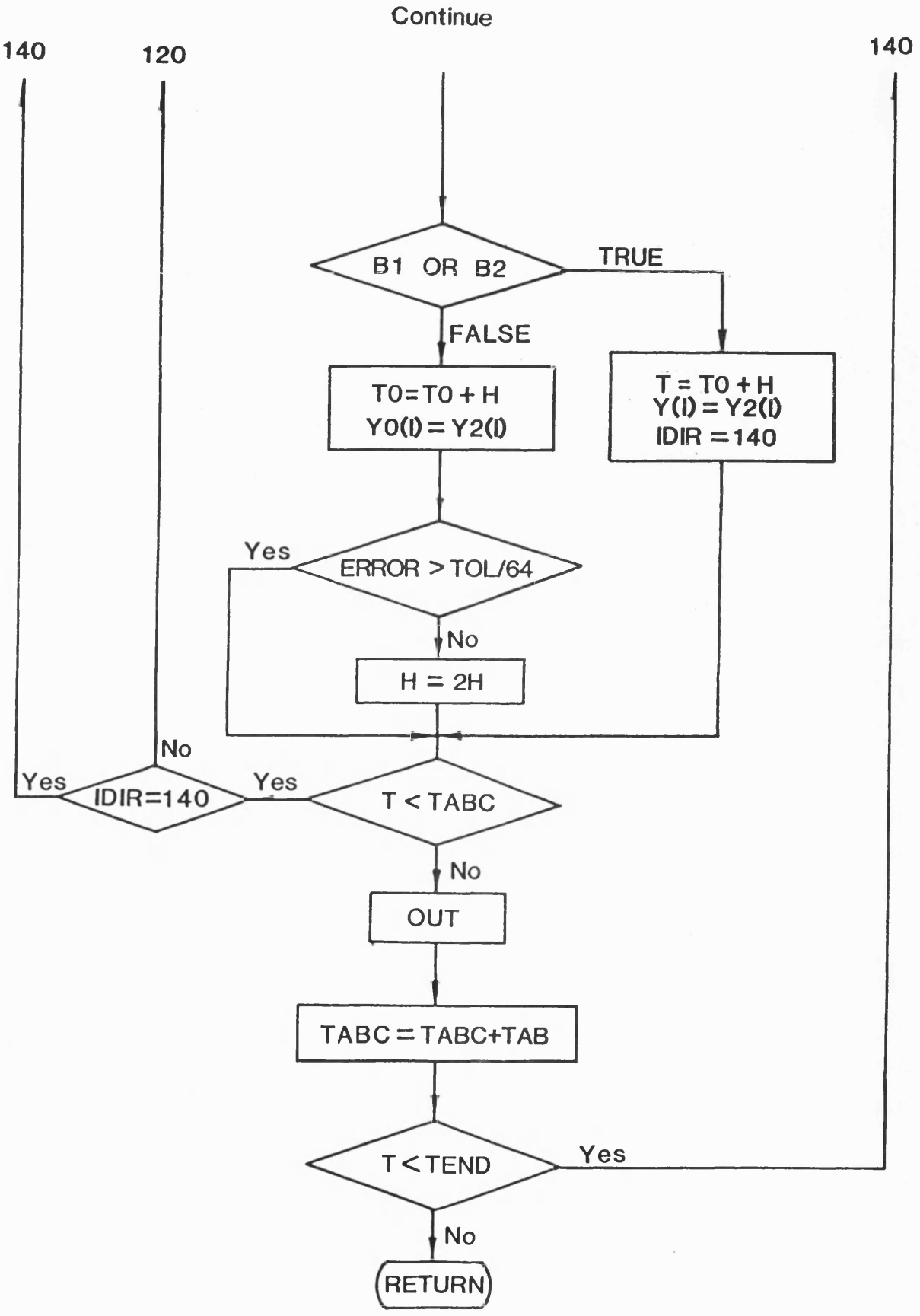


Figure 6.25 Flow chart of KUTMER integrator coding

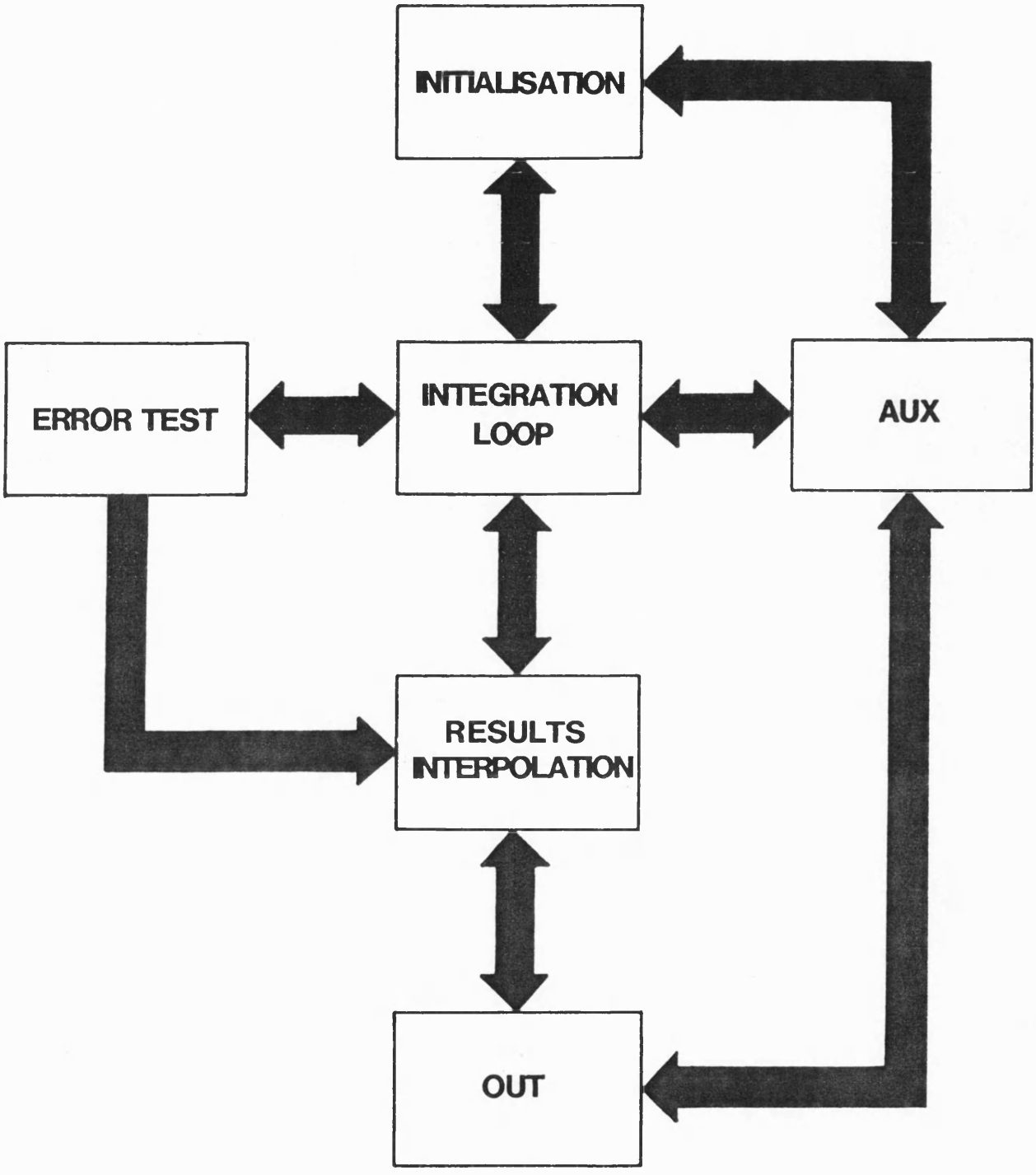
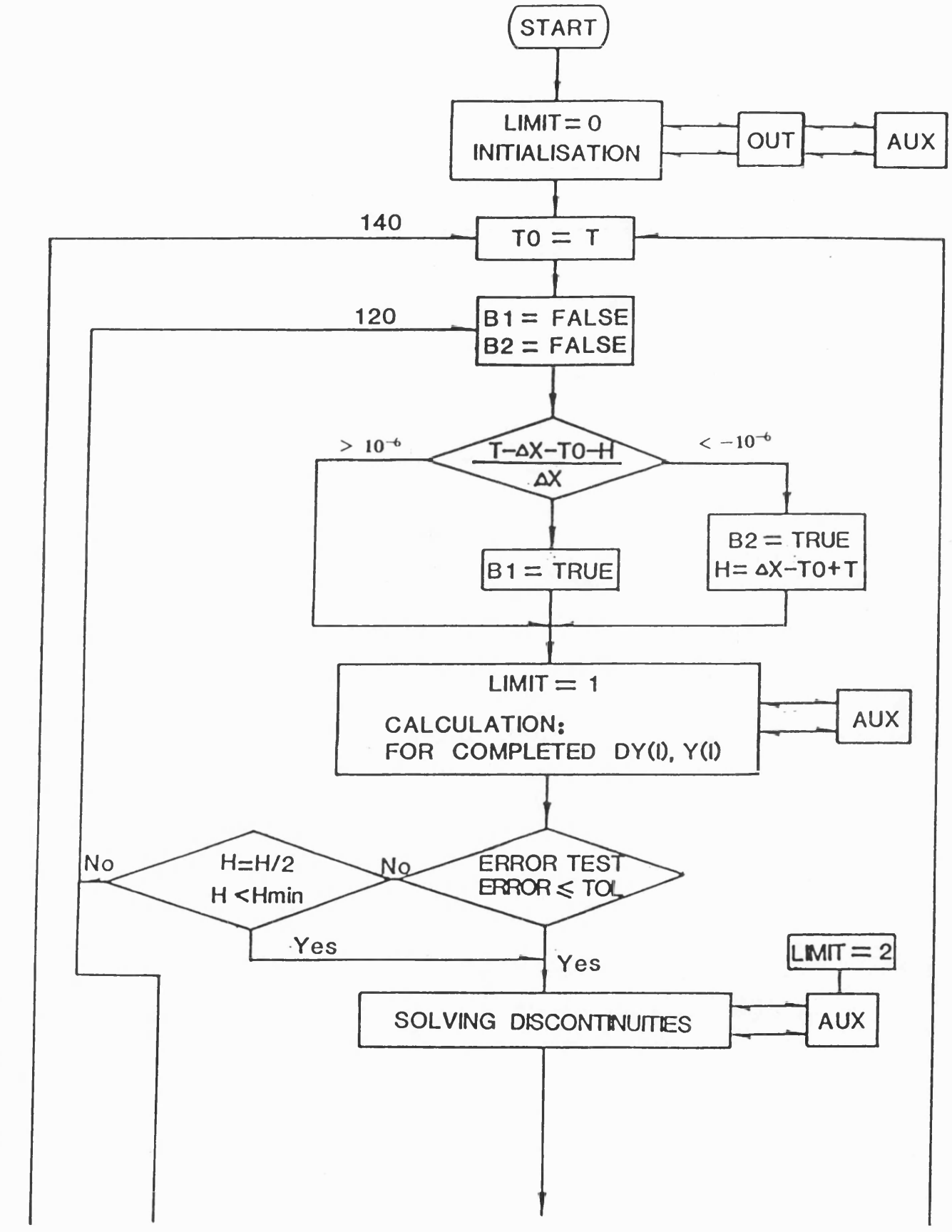


Figure 6.26 Structure of KUTMER and KUTFEH integrators


START
LIMIT = 0
INITIALISATION
OUT
AUX
140
T0 = T
120
B1 = FALSE
B2 = FALSE
> 10⁻⁶
T-ΔX-T0-H
ΔX
< −10⁻⁶
B1 = TRUE
B2 = TRUE
H= ΔX-T0+T
LIMIT = 1
CALCULATION:
FOR COMPLETED DY(I), Y(I)
AUX
No
H=H/2
H <Hmin
No
ERROR TEST
ERROR ≤ TOL
Yes
Yes
LIMIT = 2
SOLVING DISCONTINUITIES
AUX
140
120
140

Continue

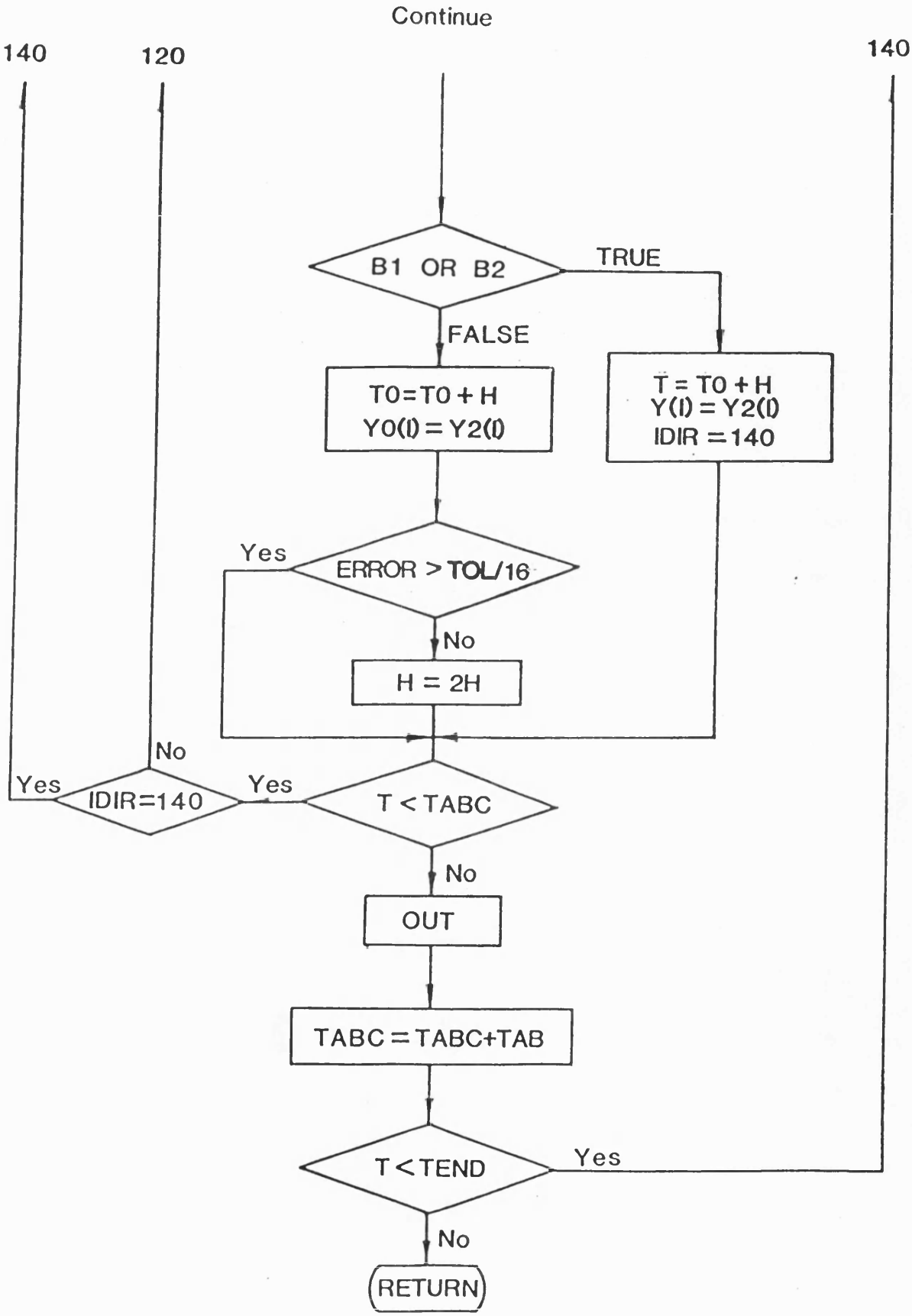


Figure 6.27 Flow chart of KUTFEH integrator coding

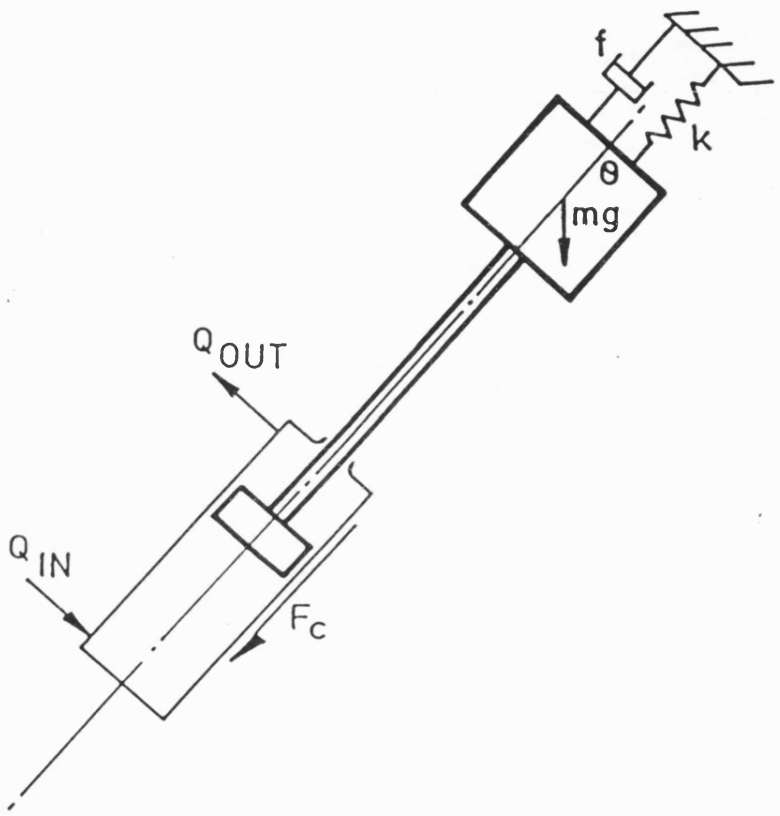


Figure 6.28 **A linear actuator**

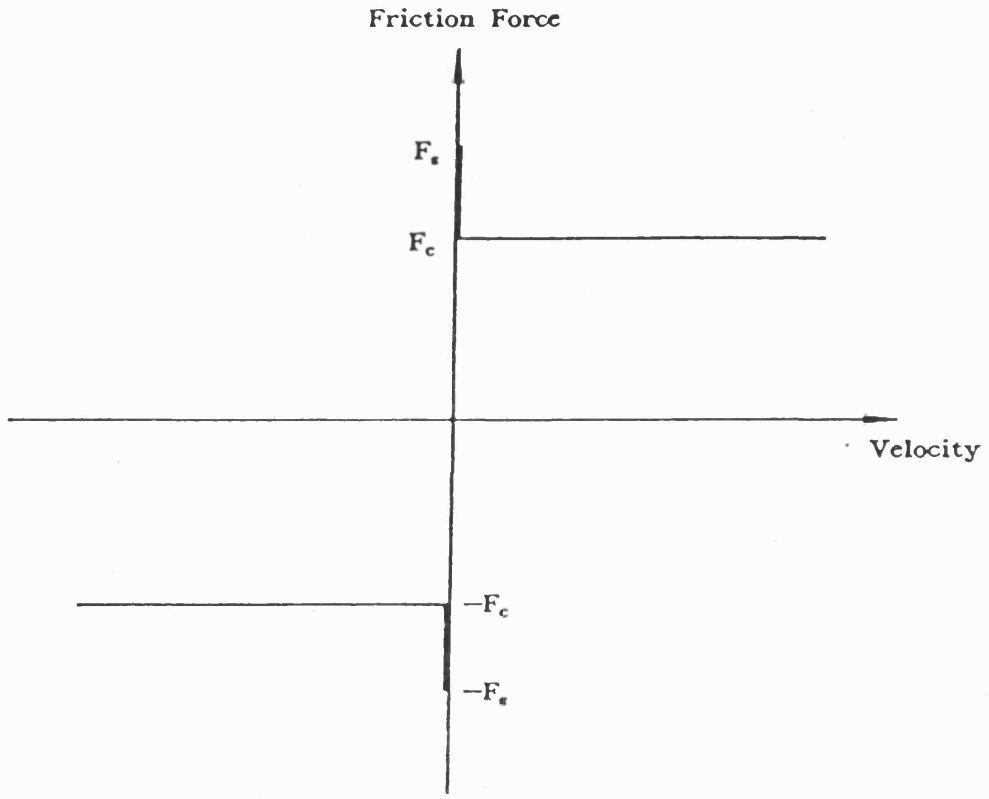


Figure 6.29 Relationship between non-linear friction force and velocity

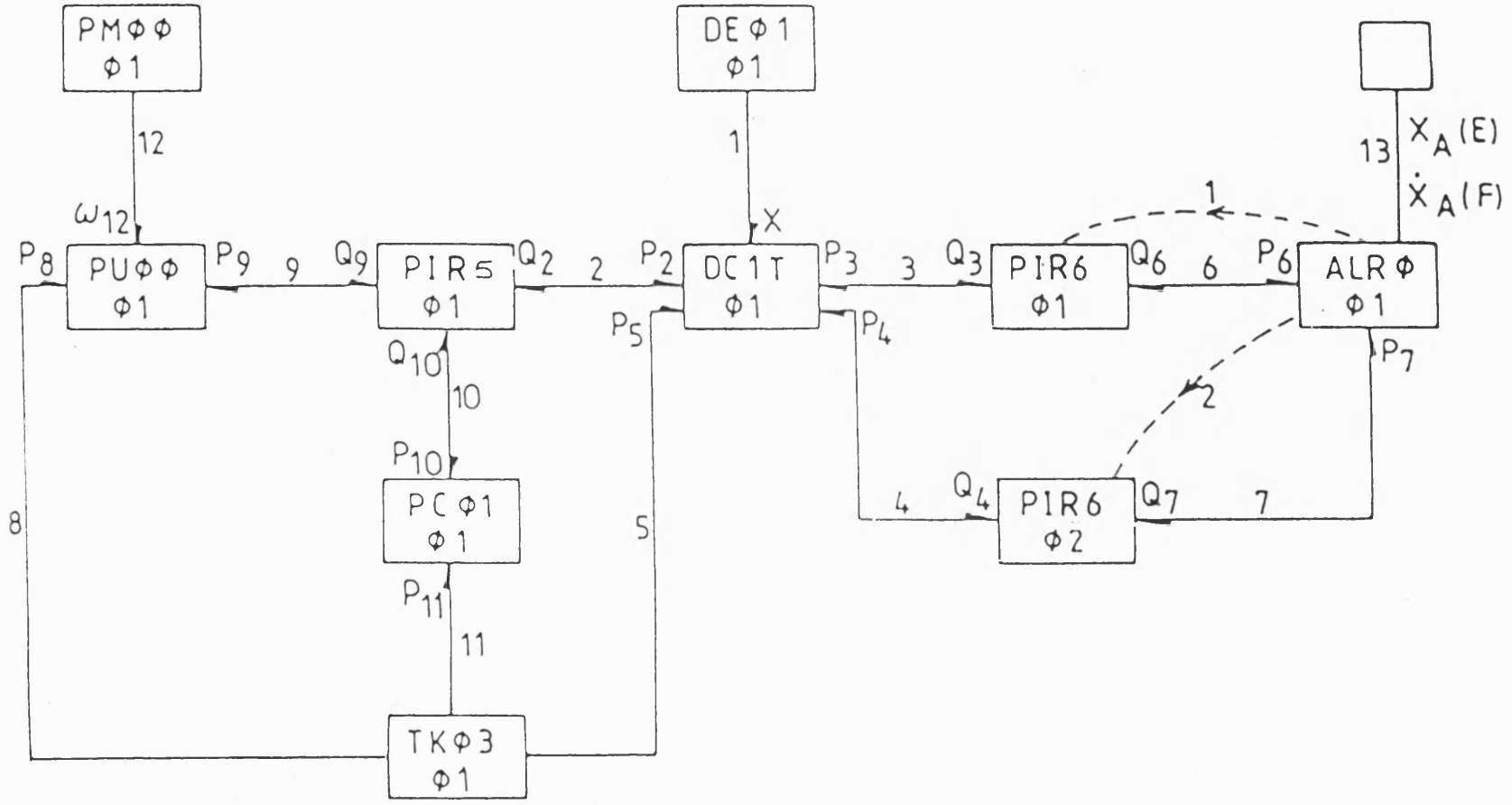


Figure 6.30 Computer linking diagram for linear actuator circuit simulation

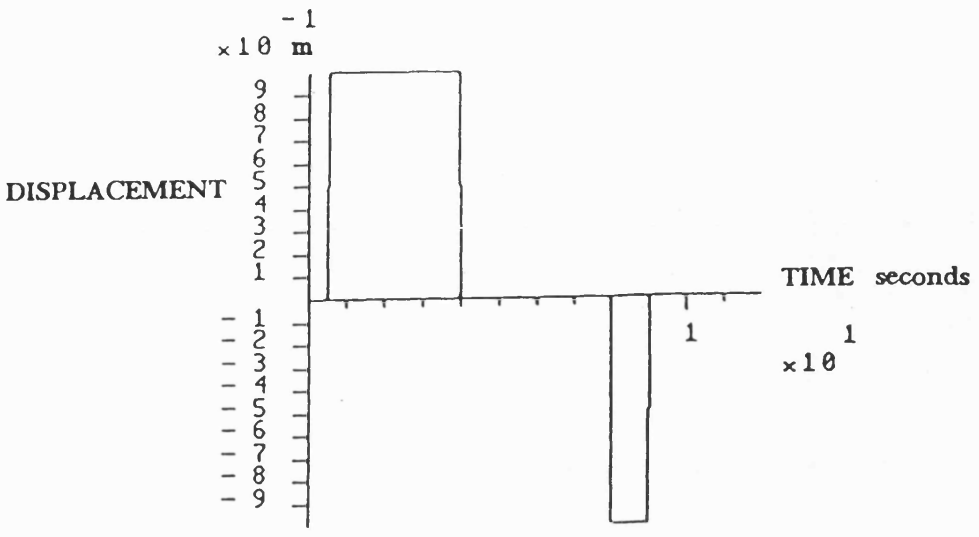


Figure 6.31 Manual position of directional control valve

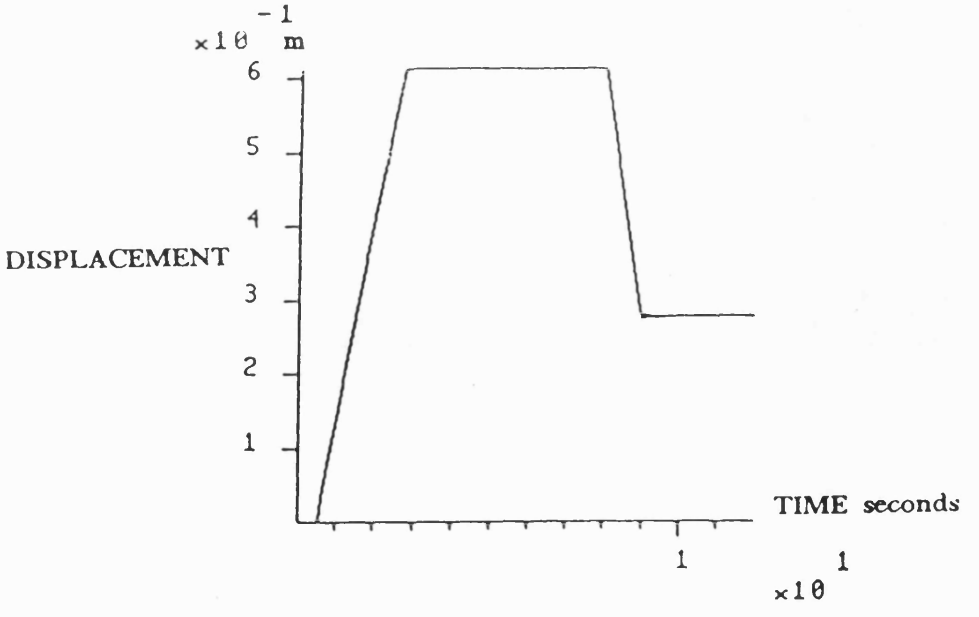


Figure 6.32 Actuator displacement

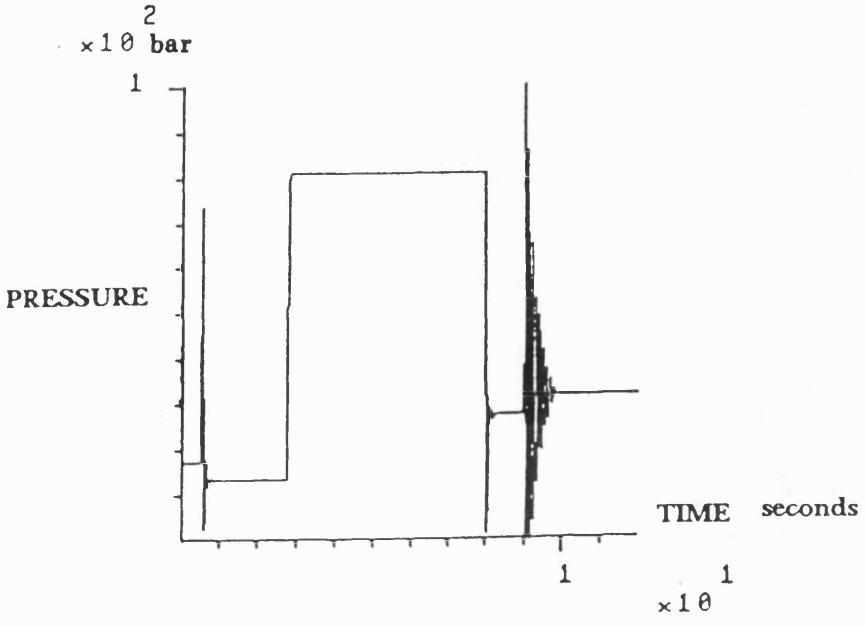


Figure 6.33 Actuator piston pressure

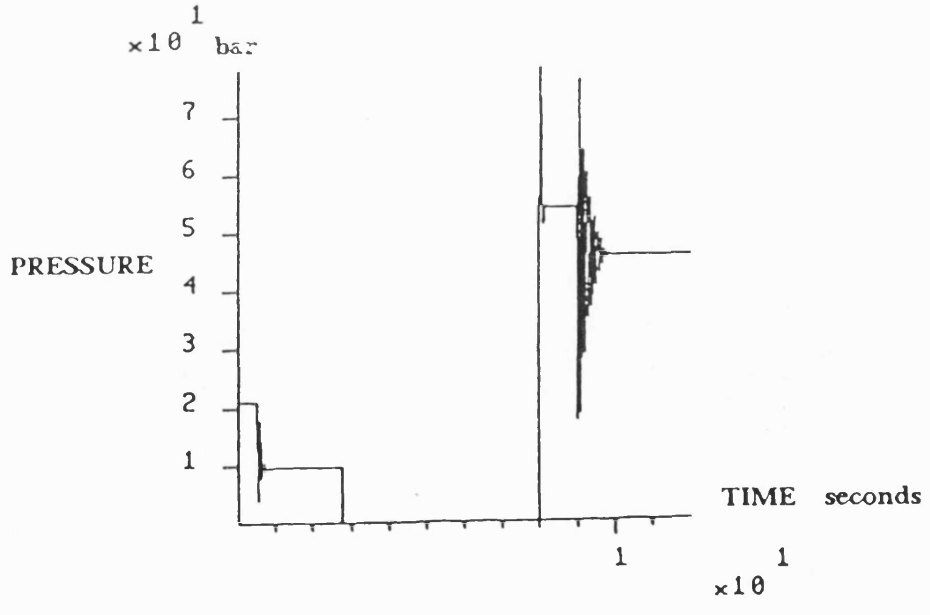


Figure 6.34 Actuator rod pressure

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

7.1 The work presented in this thesis shows that three kinds of common difficulties in the system simulation, model linking, mathematical stiffness and discontinuity, can be solved or avoided successfully by using the simulation techniques presented in the thesis.

7.2 In Chapter 2, an attempt has been made to define models as memory or non-memory models. Memory models are models which incorporate time delays or integration and which are not reliant on the states of other models to obtain their output, and it is shown that there is no difficulty in linking together memory models in any order, but with non memory models problems will arise if they are adjacent to each other. Implicit algebraic loops can arise if feedback of information occurs between non memory models. It is then shown that there are several techniques which can be applied to avoid these difficulties.

One of these is to combine several non-memory models into one large model. This unfortunately means that in a general purpose program such as HASP, the number of combination models necessary would be greatly increased and modelling problems could be large.

Sometimes a state variable can be artificially introduced into an algebraic model. This obviously will break the implicit relationship. One subset of this would be to introduce a "pseudo" state variable which has a derivative which is always set to zero. Since this state variable is calculated by algebraic equations rather than by integration, it is not a real state variable and only can be defined as a pseudo-state variable. The shortcoming of this method is that the size of the array of system state variables is increased and the computational time will be slightly increased.

7.3 In Chapter 3, an investigation of two other kinds of mathematical difficulties in the simulation of hydraulic systems, stiffness and discontinuity, has been presented. It has been shown that the main causes for the mathematical stiffness of hydraulic systems are

i) the low fluid compliance in a pipe having a small volume;
ii) the inclusion of an orifice between compressible lines;
iii) the opening of a relief or other valve for which the relation $\frac{\partial P}{\partial Q}$ is extremely low.
iv) the onset of cavitation, for the same reason.

The stiffness ratio in a simple linear actuator circuit can well be over 10^9 due to these reasons.

Since a great number of discontinuities can exist in the simulation, it can be said that the main problem of hydraulic system simulation is not that of stiffness but that of stiffness combined with multiple discontinuities. An attempt has also been made to define discontinuities as mathematical and physical discontinuities in this Chapter.

7.4 In Chapter 4, a simple iterative technique has been applied to the modelling of components to avoid the mathematical stiffness problem. The iterative procedure in these models is a half interval technique and requires the provision of flow and pressure tolerances. These tolerances are either calculated automatically as some percentage of maximum flow and pressure or directly specified by the user.

A two stage relief valve model with an iterative procedure has been developed to improve the simulation speed. This ignores the fluid compressibility in the pilot chamber. The results obtained using this approach are satisfactory as compared with those obtained using a model which incorporates compliance. The simulation speed is however very much faster using the iterative technique as compared with for the dynamic relief valve.

A similar iterative technique has been used to compute instantaneously the pressure value for a pipe with small volume. The fluid compliance in this model is assumed to be negligible and the pressure is set to a pseudo-state variable in order to overcome linking problems. It also allows the cavitation to occur when the operating pressure is less than the saturation pressure. The iterative procedure

subroutine of pipe models is automatically written by the HASP program generator for the system simulation. The relevant coding of the program generator has been added for this purpose.

This iterative technique is also used in the simulation of flow control systems. In these cases, the valve is often in close proximity to the actuator and a lengthy simulation will result if the piston in the actuator is at the valve end of its stroke. In this case iteration is used, but when the actuator moves away from the valve, the compliance of the fluid becomes significant and the fluid pressure is calculated by solving the normal state equations. A model of the pipe which is connected to an actuator and a flow control valve was developed to cope with a simulation difficulty which occurs in meter-in and meter-out flow control systems where the flow control valve is placed near to the actuator. A critical volume is set to judge when the iteration or integration algorithm is used. In both instantaneous and dynamic regions, cavitation can be shown to occur when the working pressure is less than saturation pressure and the minimum pressure value is limited to -1 bar.

7.5 A similar iterative technique has been used to model a mechanical shaft. The shaft model in Chapter 5 enables a shaft coupled power transmission to be built up separately using modular modelling technique in the HASP simulation. The problem of mathematical stiffness in the system can be avoided by the iterative procedure adopted in the shaft model. It is assumed that the shaft is rigid and takes into account the effective inertia which could vary at different times in a simulation.

It is shown that the shaft model requires the torque value from adjacent instantaneous models and returns an angular speed value. It is also shown that the angular acceleration and the torque acting on the shaft are computed by iteration and the angular speed is supplied by integration. The inertia effects of adjacent components on the shaft are considered in the shaft model through two signal links. For a shaft with a gearbox, a more complex unit model has been developed. An example of the simulation of an engine hydrostatic transmission system has also been shown.

7.6 Three single step integration algorithms have been modified and implemented in HASP to solve mathematical stiffness and discontinuity problems. It has been verified that the standard Runge-Kutta-Merson and Runge-Kutta-Fehlberg methods

would fail to simulate stiffness problems prior to the modification. It may be due to the following reasons:

i) no finite control for the change of the integration step;
ii) no control for the possible infinite loop of the error estimation of state variables.

Therefore the necessary modifications must be done for improving above two standard integration algorithms. In Chapter 6, two modifications on integration algorithms have been applied. One is an extra logical procedure which is used to aid the step control. Another is to add an integration step criterion to control the error estimation of state variables when the integration step is extremely small. In this case, a possible infinite loop of the error test of state variables can be avoided. As a result, the modified fourth order Runge-Kutta-Merson and fifth order Runge-Kutta-Fehlberg methods become suitable for solving stiff and discontinuous differential equations.

An error tolerance required in the error test of single step algorithms has been set to 10^{-5} in order to achieve reasonably accurate results. The stability of the solution can also be ensured using a very small integration time step in very stiff cases. The global error of the solution could, however, be increased and the solution accuracy would be affected if the integration time region for serious stiffness were very large.

It has also been found that the solution stability of state variables for a hydraulic system with the directional control valve (DCV) would be affected by a DCV controller model whose output is a non state variable. This is because the model of a hydraulic system with DCV could be a set of variable coefficient differential equations due to a variable input from controller model. When the displacement input of DCV controller model varies rapidly, for instance, from zero position to fully open position, the change of coefficients of differential equations might reduce the stability region of solution, and hence the solution of state variables would be unstable. One attempt to overcome this is to set a pseudo-state variable in the (controller) duty cycle model so that the displacement variable of controller model can join the error test for controlling time step size. This seemed to solve the problem but more computational time is required.

The computer modelling technique has been simplified by using the conditional statement method. It gives a simple way to code the hydraulic components described by discontinuous equations. It also shows that all discontinuity problems

including the end-stops can be modelled by IF STATEMENTS. However, these models have only been found suitable for the single step integration algorithms. It has also been pointed out that if a discontinuous model is coded by using conditional statement, a note must be put in the component attributes file COMPON.DAT, i.e. put 2 at the end of the second line of this model information section.

It is suggested that the KUTMER and KUTFEH integrators are suitable for the simulation of relatively stiff and discontinuous systems, and that the fixed time step integrator RK4 is suitable for discontinuous and non-stiff or weak-stiff systems. Typical examples of comparing the simulation speed of different integrators have been done. The effects of the stiffness and discontinuity are clearly shown.

The use of different integrators shows that the GEAR and KUTMER are suitable for general use. In the simulation of most stiff systems KUTMER could be faster than GEAR, but in some cases where the system is extremely stiff GEAR is faster than KUTMER. The present form of the GEAR integrator can only use continuous models (including artificially made continuous models) whilst KUTMER can use all of component models in current HASP model library including continuous and discontinuous models. Further work is necessary to test KUTFEH.

7.8 The HASP program generator has been partly modified in order to automatically write out simulation segments corresponding to above single step integrators.

7.9 Five models with iterative procedure including a two stage relief valve, pipes, shaft and shaft-gearbox unit, and nine general purpose models using IF STATEMENT technique have been added to the component model library.

RECOMMENDATIONS FOR FUTURE WORK

7.10 It is recommended that the research work on the multistep algorithms should be continued to eventually solve stiffness and discontinuity problems in the simulation of hydraulic systems.

1) The current version of Gear method in HASP could be modified to improve the discontinuity handling. More work could be done for the modification of the integration restart procedure.

2) It is also noted that an integrator package LSODA developed by L. Petzold and A. Hindmarsh could be implemented into HASP. This package includes the latest version of Gear and Adams methods and solves the initial value problem for stiff or nonstiff systems of first order ordinary differential equations by employing Gear and Adams methods respectively. LSODA switches automatically between stiff and nonstiff methods. This means that the user does not have to determine whether the problem is stiff or not, and the LSODA solver will automatically choose the appropriate method. Since LSODA package was developed for continuous systems, some modifications would be required for coping with discontinuous problems.

REFERENCES

[1] McArthur, C.D.
A User's Guide to CSMP, Edinburgh Catalogue No. 19.410.200
University of Edinburgh, 1972

[2] Anonymous
Continuous System Modelling Program III (CSMP III)
Program Reference Manual, Program Number 5734-XS9
IBM Corporation, 1972

[3] Green,W.L. and Speckhart,F.H.
CSMP(Continuous Systems Modelling Program).
Simulation Vol. 34, April 1980

[4] Anonymous
Advanced Continuous Simulation Language (ACSL)
User Guide/Reference Manual. Mitchell and Gauthier, 1975

[5] Smith,R.
Advanced Techniques Manual - Programming Manual
Electronic Associates Limited, Burgess Hill. Sussex. 1966

[6] Tomlinson,S.P.
HASP- Volume 1 Introductory Manual
Report No.556, School of Engineering, University of Bath, 1981

[7] Tomlinson,S.P. and Hogan, P.A.
HASP- A User Guide VAX/VMS Version
Report No.888, School of Engineering, University of Bath, 1988

[8] Tomlinson,S.P. and Hogan, P.A.
HASP-2 Volume 2. Component Model Library Manual
Report No.889, School of Engineering, University of Bath, 1988

[9] Backe,W. and Hoffmann, W.
Digital Simulation of Hydraulics (DSH), User Guide/Reference Manual.
Aachen Industrial University, West Germany, 1981

[10] Backe,W. and Hoffmann,W.
D.S.H. - Program for Digital Simulation of Hydraulic Systems
6th International Fluid Power Symposium.
University of Cambridge, UK, April 1981

[11] Amjes,G.; Levek,R. and Strfussei,D.
Aircraft Hydraulic System Dynamic Analysis,
Volume I - Transient Analysis(HYTRAN). Computer Program User Manual.
McDonnel Douglas Corporation, Report AFAPL-TR-76-43, 1977.

[12] Levek,R.; Young,B. and Strfussei,D.
Aircraft Hydraulic System Dynamic Analysis, Volume VI - Steady
State Flow Analysis (SSFAN), Computer Program User Manual.
McDonnel Douglas Corporation, Report AFAPL-TR-76-43, 1977.

[13] Levek,R. and Munroe,J.
Aircraft Hydraulic System Dynamic Analysis, Volume VII and VIII -
Transient Thermal Analysis (HYYTHA). Computer Program User Manual.
McDonnel Douglas Corporation, Report AFAPL-TR-76-43, 1977.

[14] Helsgaun, K.
DISCO. A SIMULA - Based Language for Continuous Combined
and Discrete Simulation. Simulation (SCS) Vol 35 July, 1980

[15] Tomlinson, S.P.
The Hydraulic Automatic Simulation Package-HASP
Modelling and Simulation Aspects
Ph.D. Thesis, University of Bath, 1987

[16] Hull, S.R.
The Improvement of an Automatic Procedure for the Digital Simulation
of Hydraulic Systems, Ph.D. Thesis, University of Bath, 1987

[17] Chu,Yaohan
Digital Simulation of Continuous Systems, McGraw-Hill Book Company, 1969

[18] Dugdale,S.K.
Further Development of a CAD Package for the Dynamic Simulation of Fluid Power Systems, M.Sc. Thesis. University of Bath, 1981

[19] Leung, P.S.
The Development and Testing of a Numerical Integration Method for the Digital Simulation of Fluid Power Systems, M.Sc. thesis, University of Bath, 1986

[20] Skarbeck Wazynski, C.M.
Hydraulic System Analysis by the Method of Characteristics.
Ph.D. Thesis, University of Bath, 1981

[21] Bowns,D.E. and Rolfe,A.C.
Computer Simulation as a First Step towards Computer Aided Design of Fluid Power Systems, 5th International Fluid Power Symposium, University of Durham, U.K. September 1978

[22] Gear,C.W.
Numerical Initial Value Problem In Ordinary Differential Equations, Prentice-hall, Englewood Cliffs, N.J., 1971

[23] Hall, G. and Watt, J.M.
Modern Numerical Methods for Ordinary Differential Equations.
Clarendon Press, 1976

[24] Aiken, R.C.
Stiff Computation, Oxford University Press, 1985

[25] Johnson,L.W. and Riess,R.D.
Numerical Analysis. Addison-Wesley Publishing Company, 1982

[26] J.D. Lambert
Computational Methods in Ordinary Differential Equations
John Willey & Sons, Inc. 1973

[27] Lambert, J.D. and Sigurdson, S.T.
A Generation of Multi-Step Methods for Odinary Differential Equations

Numer. Maths., 8, pp250-263, 1966

[28] Gear, C.W.
The Automatic Integration of Ordinary Differential Equations
Comput. and Comput. Mathin., 14, pp 176-179, 1971

[29] Bui, Tien D.
Solving Stiff Differential Equations in the Simulation of Physical
Systems, Simulation (August) pp. 37-46, 1979

[30] Bowns,D.E. and Rolfe,A.C.
The Digital Computation of Pressures and Flows in Interconnected
Fluid Volumes, Using Lumped Parameter Theory
4th International Fluid Power Symposium, Sheffield, April 1975

[31] Rolfe,A.C.
The Dynamic Characteristics of a Low Speed Hydraulic Motor
Ph.D. Thesis, University of Bath, 1976

[32] Rolfe,A.C.
The Computer Aided Design of Fluid Power Systems
Report No.437, School of Engineering, University of Bath, 1979

[33] Blackmore,K.Y
The Numerical Solution of Initial Value Problems in Ordinary
Differential Equations, University of Bath, 1979

[34] Caney,K.
Integration Across Discontinuities in Ordinary Differential Equations
Using Gear's Method, Report No.478, School of Engineering,
University of Bath, 1979

[35] Wang,L.M.
Modelling and Simulation of a Two Stage Relief Valve
Internal Report No.863, School of Engineering, University of Bath, 1986

[36] Bowns,D.E and Tilley,D.G.
System Dynamic and Simulation
Report No.712, School of Engineering, University of Bath, 1984

[37] Mech. Engineering Department
Mechanical Optimisation Design (In Chinese), Shanghai Jiao-tong University Press, 1981

[38] Foster, K.
Dynamic Analysis of a Two Stage Relief Valve
I.Mech.E., Oil Hydraulic Conference 1961, paper 16

[39] Zedlewski,E.J.
Modelling Techniques for the Simulation of the Two Stage Valve
M.Sc. Thesis. University of Bath, 1985

[40] Murtagh,K.F. & Turley,A.J.
Investigation of Dynamic Characteristics of a Two Stage Relief Valve
BSc Project Report, University of Bath, 1972

[41] Engineering Science Data Item Number 81039, London, Nov., 1981

[42] Bowns,D.E.
The Dynamic Characteristics of Reciprocating Engines
Ph.D. thesis, University of Bath, 1972

[43] Edge,K.E.
The Characteristics of Engine-Hydrostatic Transmission Systems
Ph.D. thesis, University of Bath, 1975

[44] McCloy,D. and Martin,H.R.
Control of Fluid Power Analysis and Design, 2nd. (revised) edition. 1980

[45] Merritt,H.E.
Hydraulic Control Systems, John Wiley, 1967

[46] Varga, R.S.
Matrix Iterative Analysis, Prentice-Hall Inc. 1962

[47] Wang, L.M.
The Application of an Iterative Optimisation Technique to the Simulation Fluid Power Systems, Report No. 866, School of Engineering,

University of Bath, 1987

[48] Magorien,V.G.
Effects of Air on Hydraulic Systems. Hydraulics and Pneumatics, 1967

[49] Bowns,D.E. & Huckvale,S.A.
A Semi-Analytic Approach to the Dynamic Simulation
of Shaft Coupled Power Transmission Systems
Report No.367, School of Engineering, University of Bath, 1976

[50] Bowns,D.E. & Worton-Griffiths,J.
The Dynamic Characteristics of a Hydraulic Transmission System
Proceedings of I Mech E. 1972, Vol 186 55/72

[51] Iyengar, S.K. & Fitch, E.C.
A Systematic Approach to the Analysis of Complex Fluid Power Systems.
4th International Fluid Power Symposium, April 1975, Paper No. A2.

[52] Paynter,H.M.
Analysis and Design of Engineering Systems
M.I.T. Press, Cambridge, Mass. 1960

[53] Dransfield, P. & Barnard, B.
Dynamic Modelling of Hydraulic Control Systems. World Conference in
Industrial Tribology, New Delhi, India, December 1972

[54] William S. Dorn, Daniel D. McCracken
Numerical Methods with Fortran IV Case Studies
John Willey & Sons, Inc. 1978

[55] A.M. Cohen
Numerical Analysis, McGraw-Hill Book Company (UK) Ltd. 1973.

[56] S.C. Mathewson
Simulation Program Generators, Simulation (December) pp.181-189, 1974

[57] Smith,J.M.
Mathematical Modeling and Digital Simulation for Engineers
Scientists. John Wiley and Sons, 1977

[58] Chai, Augustine S.
A Modified Runge-Kutta Method, Simulation May 1968. pp. 221-223

[59] Chai, Augustine S.
A Fifth-order Modified Runge-Kutta Integration Algorithm
Simulation January, 1972. pp.21-27

[60] Chai, Augustine S.
Modified Merson's Integration Algorithm which Saves Two
Evaluations at Each Step, Simulation March, 1974. pp. 90-92

[61] Chai, Augustine S.
Comments on the Runge-Kutta-Merson Algorithm
Simulation August, 1970. pp. 89-91

[62] Merson, R.H.
An Operational Method for the Study of Integration Process
Pro. Symp. Data Processing, Weapons Research Establishment,
Salisbury, S. Australia, 1957

[63] Ray, Asok
Dynamic Modeling and Simulation of a Two Stage Relief Valve
Simulation November, 1978. pp. 167-172

[64] Inoue, R.
The Simulation of Pilot-Operated Relief Valves
The BFPR Journal. 1980, Oklahoma State University, USA, pp225-228

[65] Iyengar, S.K.R.
Static and Dynamic Performance Degradation of Compound Relief Valves
due to Contaminant Exposure, 5th International Fluid Power Symposium,
Durham, England, 1978

[66] Spohr, H.
Pilot-Operated 2-way, 2-position Poppet Valves with Electronic
Surveillance - Their Construction and Use, 5th International Fluid
Power Symposium, Durham, England, 1978

[67] Zeiglar, B.P.
Theory of Modelling and Simulation, John Wiley & Sons, Inc. 1976

[68] Tomlinson,S.P. & Bowns,D.E.
The Hydraulic Automatic Simulation Program (HASP)
Hydraulic Engineering Conference, RNEC Manadon, 1982

[69] Bowns,D.E., Tomlinson,S.P. and Chapple,P.J.
The Simulation of Thermal Transient Effects Using the Hydraulic
Automatic Simulation Package,
7. Aachener Fluid Technisches Kolloquium, W. Germany, 1986

[70] Bowns,D.E.; Tomlinson,S.P. and Hull,S.R.
Application of an Hydraulic Automatic Simulation Package to the
Design of Fluid Power Systems. Beijing Fluid Power International
Symposium, P.R. China, October 1984

[71] McCandlish, D. and Dorey, R.E.
The Mathematical Modelling of Hydrostatic Pumps and Motors
Proc. Instn. Mech. Engrs. Vol 198B No 10, 1984. pp 165-174

[72] Ayres,J.L.
New CAD Package for FLuid Power Transmission, CME April 1987, pp26-27.

[73] Chapple,P.J.
Computers Help Hydraulics Designers, Engineering July 1986, pp546-547.

[74] Krus, P.
The Simulation of Fluid Power Systems with Complex Load Dynamics
Dept. of Mechanical Engineering, Linkoping University, Sweden. 1986

[75] Krus, P.
HOPSAN, Users Manual
Dept. of Mechanical Engineering, Linkoping University, Sweden. 1986

[76] Bowns,D.E.; Tomlinson,S.P. and Dugdale,S.K.
Progress towards a General Purpose Hydraulic System Simulation
Language, 6th International Fluid Power Symposium, Cambridge, 1981

[77] Bowns,D.E.; Tomlinson,S.P.
Performance Prediction for Fluid Power Systems,Power,May 1985 pp114-116

[78] Hull,S.R. and Bowns,D.E.
The Development of an Automatic Procedure for the Digital Simulation of Hydraulic Systems, I Mech E Conference CAD in High Pressure Hydraulics. Nov. 1983

[79] Bowns,D.E.; Tomlinson,S.P.
The Role of Simulation and Experiment in Hydraulic Systems Design, The Congress of Oil Hydraulics, Bologna, Italy. 1983

[80] Edge,K.A. and Leahy,J.C.
Digital Computer Simulation As an Aid in Improving the Performance of Positive Displacement Pumps with Self-Acting Valves
Proc Instn Mech Engrs Vol 198 No 14, 1984. pp 267-274.

[81] Dorey, R.E.
Hydrostatic and Split Power Transmissions and Their Application to the City Bus. PhD. Thesis, 1983. University of Bath

[82] Dorey, R.E.
Computer Simulation of Control Systems, Joint Inst. M.C(uth)/SERC workshop on "Computer Aided Control System Design". University of Sussex, Brighton, September 1984

[83] Bowns, D.E.; Hull, S.R. and Tomlinson, S.P.
An Automatic Generation Process for Dynamic Simulation Programs
Trans. Scy Computer Simulation, USA, April 1984

[84] Caplen, M.J.S.
The Development and Testing of a New Integration Method for the Solution of Stiff Differential Equations Arising in the Digital Simulation of Fluid Power Systems, Ph.D. thesis, to be published, University of Bath

[85] Carnahan,B.; Luther,H.A. and Wilkes,J.O.
Applied Numerical Methods. John Wiley and Sons, 1969.

[86] Pippenger,John J.
Hydraulic Valves and Controls, Marcel Dekker, Inc. 1984

[87] Balfour,A. and Marwich,D.H.
Programming in Standard FORTRAN 77, Heinemann Educational Books. 1979

[88] Streeter, V.L.
Fluid Transients, McGraw Hill International Book Company, 1978.

[89] Thoma, J.U.
Introduction to Bond Graphs and Their Applications
Oxford, Pergamon Press, 1975.

[90] Fox, J.A.
Hydraulic Analysis of Unsteady Flow in Pipe Networks. 1977

[91] Bowns, D.E. and Worton-Griffiths, J.
The Effect of Air in the Fluid on the Operating Characteristics
of a Hydraulic Transmission, 3rd International Fluid Power
Symposium, Turin, Italy. 1973.

[92] Magorien, V.G.
Effects of Air on Hydraulic Systems. Hydraulics and Pneumatics, 1967.

APPENDIX A

MODIFICATIONS OF THE HASP PROGRAM GENERATOR

1. **A Description of the HASP Program Generator.** The program generator is the core of the HASP simulation package. Its main purpose is to write out a few FORTRAN control segments which link existing component models and numerical integrator to form a complete simulation program. Before writing FORTRAN control segments, It has to complete two functions below.

i) analyse the circuit data required as input and check for errors or model compatibility.

ii) arrange the arbitrarily ordered models into an acceptable calling sequence of component subroutines in the system model segment AUX.

The functions of the program generator are shown in figure 1.2.

2. **Basic structure of the program generator** is shown diagrammatically in figure A.1. The program generator main segment PGMAIN which is listed at level 1 in figure A.1 consists of four segments decribed below.

i) **Circuit data input segment PGIN.** This checks the validity of defined circuit data.

ii) **Circuit data analysis segment PGCOMP.** This analyses the circuit details in terms of the input circuit data file and then provides a calling list or error diagnostics.

iii) **FORTRAN file writing segment PGOUT.** This writes four FORTRAN control segments, MAIN.FOR, CONTRL.FOR, AUX.FOR and

OUT.FOR, to form a majority of the HASP simulation program as shown in figure A.2. Some contents of these FORTRAN control segments will be different if different numerical integrator is chosen.

iv) Model selector writing segment PGSEL. This segment writes a model selector file CAD.OPT. The selector file, CAD.OPT, contains the selected component models, integration subroutine and other required subroutines for forming a simulation program file CAD.EXE.

3. A correct calling sequence of above segments in the program generator PGMAIN is:

```
CALL PGIN
CALL PGCOMP
CALL PGOUT
CALL PGSEL
```

These segments are listed in level 2 of the structure diagram (figure A.1) and other subroutines pertaining to the generator segments are listed in level 3 of the diagram. Figure A.3 shows a flow diagram for current HASP program generator main segment PGMAIN. A detailed description about these segments and relevant subroutines were given by Hull[16].

4. **Model attribute file COMPON.DAT related with program generation.** When the segment PGCOMP of the program generator is used to analyse the validity of circuit data, a model attribute file COMPON.DAT is interrogated. The file COMPON.DAT includes such information as link and signal details as well as the numbers of state variables, real and integer parameters etc. Consider a component with the hypothetical name COMP, the model attribute details for COMP in COMPON.DAT will be of the form[15]:

```
COMP
ABCDEFGHIJK
```

where:

"COMP" is the subroutine name. In the second line, "A" is the minimum number of external links which may be attached to the model. "B" is the maximum number of external links. "C" is the number of internal links. "D" is the number of signals. "E" is the number of state variables used in the model. "FG" is a two digit number defining the number of real constants used by the model subroutine and "HI" is a two digit number defining the number of integer constants. "J" is a yes (Y) or no (N) answer to the quenstion "Does time (T) appear explicitly in the calculation subroutine argument list?". Time (T) is the simulation time. "K" is an integer defining the response to the question enquiring whether the integer variable LIMIT appears in the subroutine argument list. If no "K" is set to 0. If yes but the state variables will not be reset from the link variables (EFFORT or FLOW) to state variables Y(I) at the bottom of the AUX.FOR subroutine "K" is set to 1. If they are reset "K" has to be set to 2.

For instance, a linear actuator model ALR0. The required attribute details (two lines) for ALR0 in COMPON.DAT is of the form:

```
ALR0
221222904N2
```

This shows that the model ALR0 has two external links, one internal link, two signals, two state variables, twenty-nine real constant parameters, four integer constants and no time (T) is required in the argument list of the model subroutines. It also shows that two state variables of ALR0 need to be reset at the bottom of the system model subroutine AUX.FOR because "K"is set to 2.

When the program generation stage is initiated, the details of link, signal, state variable and parameter number for every model used in circuit must be checked to ensure that they are compatible with adjoining models and accepted by the program generator. If checking is successful, these details in COMPON.DAT are then used to write the FORTRAN segments by the program generator. Otherwise, the relevant diagnostic error message appears on the terminal to warn the user that the model is unacceptable. A full description about COMPON.DAT is given by Tomlinson[15].

5. **Modification of the Program Generator for Writing Iterative Procedure Subroutines ITSEL and ITERi of Pipe Model PIA5 or PIA6.** The pressure in the pipe model PIA5 or in the instantaneous pressure region of PIA6 is computed instantaneously by an iterative procedure (see paragraph 4.23). This iterative procedure is carried out by calling two modular subroutines **ITSEL** and **ITER** in the model PIA5 or PIA6 (see figure 4.16). Since the adjacent models of PIA5 (PIA6) may be different in different circuit, the calling sequence of these models to compute flow error in an iterative process of the pressure value may be different. Hence the subroutines **ITSEL** and **ITER** would require to be written manually if the program generator were not modified.

As stated previously, the main purpose of the program generator is to write FORTRAN source files required to form a simulation program. Therefore, once relevant modifications are done, these iterative subroutines can be written automatically by the HASP program generator. If the model PIA5 (or PIA6) is employed more than once (but not more than ten) in the same circuit, for instance **i** times, the **i** subroutines **ITER1**, **ITER2**,...., **ITERi** will be written for PIA501, PIA502, ..., PIA50i respectively. A control subroutine **ITSEL** is also required and written for selecting different **ITERi** when the model PIA50i is called in the simulation of the system.

6. **The rudiments of writing iterative procedure subroutines.** The HASP program generator has been modified by the author for writing ITSEL and ITERi. The use of PIA5 or PIA6 will be identified by the segment PGIN of the program generator from the model attribute details in COMPON.DAT, and then the FORTRAN source files of ITSEL and ITERi will be automatically written by the output segment PGOUT. This automatic procedure is carried out by means of the model attribute information specified by the model writer and the circuit input data. For instance, the model attributes information of pipe model PIA5 is specified in COMPON.DAT and is given below,

PIA5
180011203N3

The value "K" (see paragraph 4) at the end of the second line is set to 3. This command means that: (i) the state variables will be reset from the link variables

(EFFORT or FLOW) to state variables Y(I) at the bottom of AUX.FOR; (ii) ITSEL and ITERi subroutines will be written automatically by the program generator.

As an example, the input data of a circuit to be simulated is shown as follows:

```
10
:
:
PIA501  4  5
PIA502  7  8
:
:
```

It is noted that the model PIA5 is used twice in the circuit simulation. Therefore, two iterative subroutines ITER1, ITER2 and one seletion subroutine ITSEL which includes two calling statements as listed

```
IF(N.EQ.1)CALL ITER1(...)
IF(N.EQ.2)CALL ITER2(...)
```

will be written down automatically by the program generator.

7. **Modifications in program generator.** In order to write ITSEL and ITERi subroutines, following modifications have been done in the program generator:

(i) In the output segment PGOUT, a section for writing the source coding of the subroutine ITSEL is added. If the model PIA5 is used **N** times (N=1,10), a full FORTRAN file ITSEL is written and its contents depends on the N number of times of using PIA5 in the same circuit. For instance, PIA5 is used twice in a circuit, ITSEL is then written as:

```
C
C %%%%
```

(i) In the main segment PGMAIN, RK4 and KUTMER integrators are put into the menu of integration algorithms which is shown below.

1. GEAR (DEFAULT)
2. HINDMARSH GEAR
3. FOURTH ORDER EXPLICIT RUNGE-KUTTA --- RK4
4. FIRST ORDER METHOD
5. IMPLICIT RUNGE-KUTTA
6. KUTTA-MERSON --- KUTMER

(ii) One writing section is added into the output segment PGOUT to write a following writing section for CONTRL.FOR file when the number is set to 3. i.e.

```
224   WRITE(6,225)
225   FORMAT(' TYPE IN TIME STEP FOR R-K METHOD IN SECONDS: ')
      READ(5,220,ERR=224)HT
      IF(HT.LE.0.D0)GO TO 224
      IF(MRSP.EQ.7)GO TO 300
```

This is because the RK4 integrator is a fixed time step algorithm.

(iii) Four writing statements are also added into the output segment PGOUT to rewrite the DIMENSION and CALL statements for MAIN.FOR file by means of the number chosen by users, 3 or 6, i.e.

```
            :
      DIMENSION DOT( N),Y( N),YC( N),Y1( N)
            :
      CALL RK4(N,T,TEND,TAB,Y,DOT,YC,Y1,AUX,OUT)
            :
```

or

```
            :
```

```
DIMENSION DOT( N),Y( N)
          :
CALL KUTMER(N,T,TEND,TAB,Y,DOT,TOL,AUX,OUT)
          :
```

(iv) Another two writing statements are added into the selector segment PGSEL to write one of following statements for the selection file CAD.OPT due to above selected number 3 or 6.

```
[HASP.COMPON]RK4,-
```

or

```
[HASP.COMPON]KUTMER,-
```

9. **Modification of the Program Generator for Increasing the Maximum Number of Component Models Used in a System Simulation.** Initially the maximum number of components used in a system simulation was twenty-five although the maximum number of state variables allowable was ninety-nine. Now the maximum number of components used in a system has been modified to 50 by increasing the size of the array dimensions from 25 to 50 in the program generator segments.

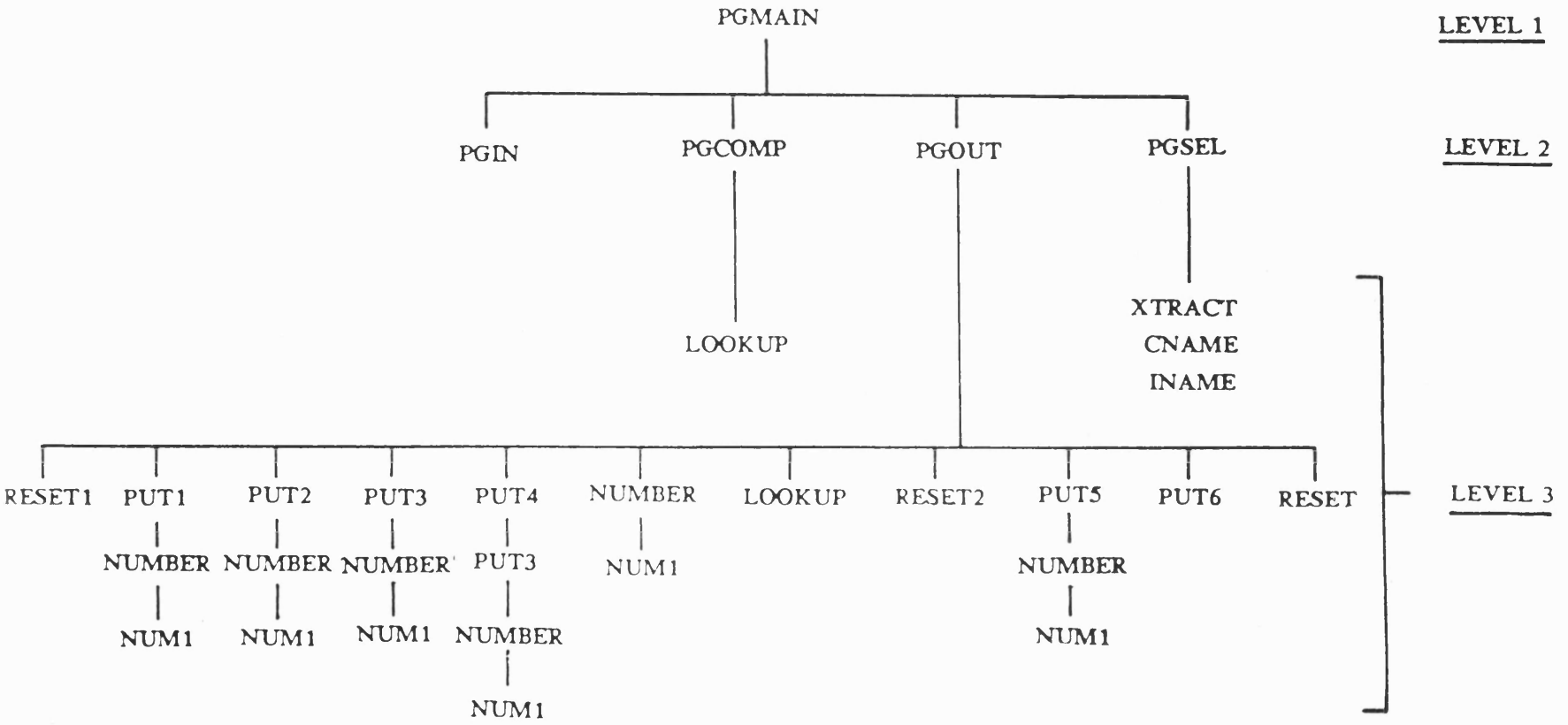


Figure A.1 Basic structure of the program generator

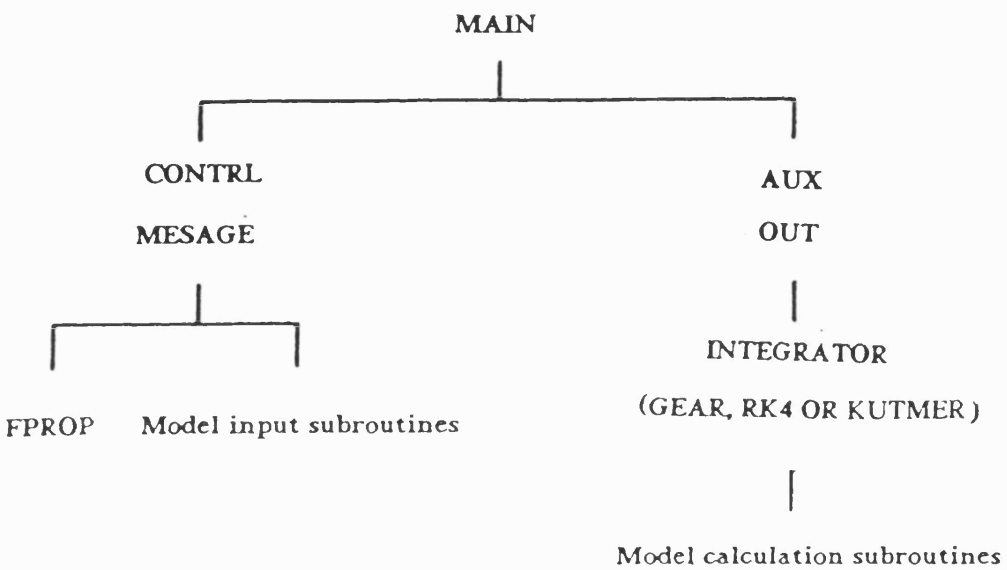


Figure A.2 An overlay tree for a HASP simulation program

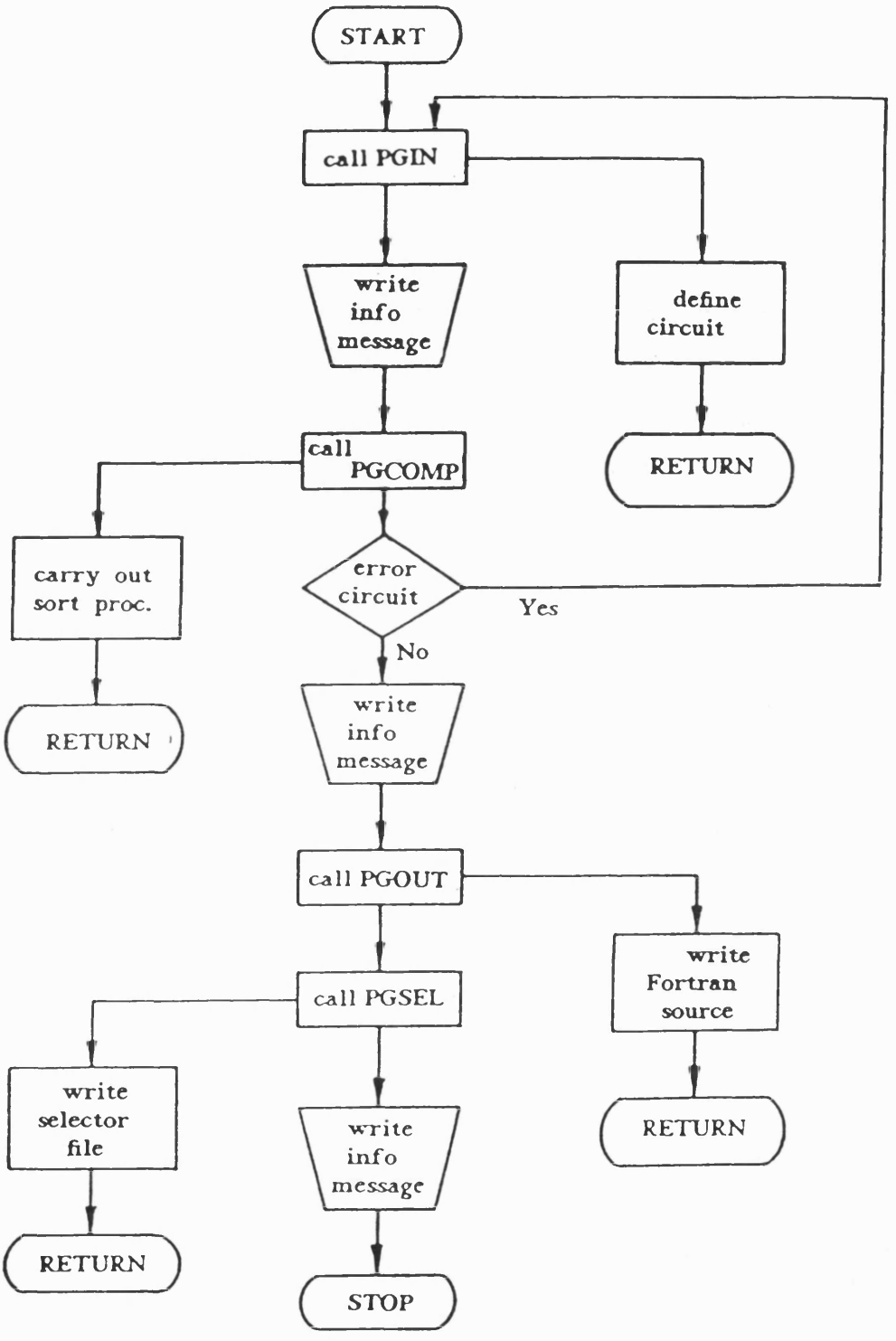


Figure A.3 Flow diagram for program generator main segment

APPENDIX B

DOCUMENTATION OF COMPUTER MODELS

USING ITERATIVE PROCEDURE

CONTENTS

N.B. These models are available in the HASP model library.

PCA1 - A TWO STAGE RELIEF VALVE MODEL WITH ITERATIVE PROCEDURE

1. Introduction.

PCA1 simulates the behaviour of a two stage relief valve. The model does not take into account the fluid compressibility, fluid friction and fluid inertia in the pilot stage chamber and the transient flow force acting on the main spool and pilot poppet. In order to avoid the mathematical stiffness problem caused by small fluid volume in pilot chamber, an iterative procedure is adopted to instantaneously compute the pressure in pilot chamber. In this model the inlet pipe linked with the two stage relief valve is not considered, but it will be used as an external component model in the simulation of a two stage relief valve. A schematic of a two stage relief valve is shown in figure B-1. The design symbol for the component is shown in figure B-2.

The model can only be used for simulations using Runge-Kutta or Kutta-Merson integration subroutines.

2. Model Assumptions.

(1) the poppet valves have chamfer no their seats.
(2) the compressibility of the oil in pilot and downstream chambers is neglected.
(3) no radial forces exist on either of the poppet valves lateral forces acting on the valve are ignored.
(4) transient flow forces acting on poppet valves are ignored.
(5) gravity effects are neglected.
(6) flow is irrotational.

3. Nomenclature.

t - time
Q - flow rate
P - pressure
P_{cr} - cracking pressure of a two stage relief valve
β - fluid bulk modulus
C_d - discharge coefficient of the orifice
f - viscous coefficient
C_L - amending coefficient
K - spring stiffness
M - mass
D - diameter
d_o - diameter of control orifice
α - angle of poppet valves
A - area
V - fluid volume
X, Y - displacements of main and pilot poppet valves
v_x, v_y - velocities of main and pilot poppet valves
ρ - density of the oil
Δf - flow iterative error tolerance
Δp - pressure iterative error tolerance

Suffixes

1 - main stage
2 - pilot stage
3(c) - contraction cross section
L - orifice
e - effective value
p - spring chamber of main stage

4. User defined parameters

(1) cracking pressure in bar
(2) mass of pilot poppet in kg
(3) spring stiffness of pilot poppet in N/m
(4) viscous coefficient of pilot poppet in Ns/m
(5) diameter of pilot stage poppet seat in m
(6) area of pilot stage poppet seat in m^2
(7) angle of pilot poppet face in degree
(8) spring preload of main stage in N
(9) mass of main poppet in kg
(10) spring stiffness of main stage in N/m
(11) viscous coefficient of main stage Ns/m
(12) diameter of inlet port of main stage in m
(13) area of inlet port of main stage in m^2
(14) angle of main poppet face in degree
(15) diameter of main spring chamber in m
(16) area of main spring chamber in m^2
(17) clearance between main poppet and valve cover in m
(18) viscous length of main stage in m
(19) diameter of control orifice in m
(20) initial pressure value of pilot stage in bar
(21) maximum flow rate through pilot stage in m^3/s
(22) maximum pilot chamber pressure in bar

5. Model Equations.

The mathematical equations of this model comprise one iterative procedure for the pressure of pilot chamber and two motion equations for main and pilot poppets.

Iterative procedure for pilot pressure.

The iterative process commences by estimating a value for pressure P_2 using the equation:

$$P_2^{(k)} = \frac{P_{2H} + P_{2L}}{2} \qquad \ldots\ldots (B.1)$$

where

$P_{2H} = P_{IN}$, the inlet pressure

and $P_{2L} = P_{OUT}$, the drain line pressure

k - the number of iteration (initial value is 1)

i) The pressure $P_2^{(k)}$ obtained from above equation is used to calculate the input and output flow rates of pilot stage. i.e.

$$Q_{in} = Q_L + Q_{VX}$$

$$= K_{or} \sqrt{(P_{IN} - P_2^{(k)})} + A_p v_x \qquad \ldots\ldots (B.2)$$

$$Q_{out} = Q_2 + A_2 v_y$$

$$= \pi C_d D_2 Y Sin(\alpha_2) \sqrt{2(P_2^{(k)} - P_3)/\rho} + A_2 v_y \qquad \ldots\ldots (B.3)$$

ii) flow and pressure errors are estimated by:

a) flow error: $\Delta Q = Q_{in} - Q_{out}$

b) pressure error: $\Delta P = P_2^{(k)} - P_2^{(k-1)}$

iii) Halving iterative interval for the estimation of value of P_2, if necessary.

a) If the flow error between the flows is within a set tolerance Δf, i.e.

$$| \Delta Q | \leqslant \Delta f$$

the pressure value chosen $P_2^{(k)}$ is considered acceptable, i.e.

$$P_2 = P_2^{(k)}$$

and then the iteration for pressure P_2 at this time point is finished.

b) If $| \Delta Q | > \Delta f$ but $| \Delta P | \leqslant \Delta p$, as shown in figure 4.3(b), it shows that although the flow error is outside the flow tolerance Δf the estimated pressure $P_2^{(k)}$ has been near the real value P_2, i.e. pressure error obtained has been within a set tolerance Δp. In this case the estimated pressure value P_2^k may be considered acceptable, i.e.

$$P_2 = P_2^{(k)}$$

and then iterative process is finished.

c) Otherwise the calculations are repeated using a new halved iterative interval $[P_{2L}, P_{2H}]$, when

$$P_{2L} = P_2^{(k)} \text{ and } P_{2H} = P_{2H} \text{ for } Q_{IN} > Q_{OUT} \qquad \ldots\ldots \text{(B.4)}$$

or

$$P_{2L} = P_{2L} \text{ and } P_{2H} = P_2^{(k)} \text{ for } Q_{IN} < Q_{OUT} \qquad \ldots\ldots \text{(B.5)}$$

The iteration process continues until an acceptable value for P_2 is obtained.

Determination of iterative tolerances. In order to compute the pilot chamber pressure instantaneously by iteration, one flow error test and a pressure error test are required. The error tolerances employed in two error tests can be specified by the user or calculated in terms of maximum pressure and flow rate values supplied by the user using following tolerance proportional functions.

Flow rate tolerance is given by:

$$\Delta f = Q_{max}/100000$$

Pressure tolerance is given by:

$$\Delta p = P_{2max}/1000$$

Motion equation for main poppet valve

$$M_1 \frac{d^2X}{dt^2} + f_1 \frac{dX}{dt} + K_{e1} X = P_{IN} A_1 - P_2 A_p - F_1 \qquad \ldots\ldots (B.6)$$

where equivalent spring stiffness (see paragraph 4.6)

$$K_{e1} = K_1 + \pi C_d D_1 Sin(2\alpha_1)(P_{IN} - P_{OUT})$$

Motion equation for pilot poppet valve

$$M_2 \frac{d^2Y}{dt^2} + f_2 \frac{dY}{dt} + K_{e2} X = P_2 A_2 - P_{cr} A_2 \qquad \ldots\ldots (B.7)$$

where equivalent spring stiffness

$$K_{e2} = K_2 + \pi C_d D_2 Sin(\alpha_2)(P_2 - P_{OUT})$$

Consideration of end stops. The motion of the poppet or spool between the end stops ($0 < X < X_{max}$ and $0 < Y < Y_{max}$) is described by the equations of motion shown in equations (B.1) and (B.2). The mathematical end stop regions are defined for $Y > Y_{max}$, $X > X_{max}$ and $Y < 0$, $X < 0$, but in the practical conditions the poppet velocity must be zero when the poppet hits the end stop. Hence the mathematical model of the end stop for a two stage relief valve needs to be established so that the complete motion characteristics of the main and pilot poppets can be described.

Actually the model of the end stop can be described by non-linear physical constrained functions and solved easily using logical conditional statements. For a pilot poppet the mathematical model of the end stop is expressed using the following non-linear physical functions:

$$Y = \begin{cases} 0 & \text{if } Y < 0 \\ Y_{max} & \text{if } Y > Y_{max} \end{cases} \qquad \ldots\ldots (B.8)$$

$$v_y = 0 \text{ and } \dot{v}_y, \quad \text{if } Y < 0 \text{ or } Y > Y_{max} \qquad \ldots\ldots (B.9)$$

similarly, for the main poppet the mathematical model of the end stop is given by:

$$X = \begin{cases} 0 & \text{if } X < 0 \\ X_{max} & \text{if } X > X_{max} \end{cases} \qquad \ldots\ldots (B.10)$$

$$v_x = 0 \text{ and } \dot{v}_x, \quad \text{if } X < 0 \text{ or } X > X_{max} \qquad \ldots\ldots (B.11)$$

6. Model Linking.

The model receives inlet and outlet pressure as input and supplies flow to adjoining models via two external links. The model also has 3 internal links. Two internal links are designed for displaying the displacements and relevant velocities of the main and pilot poppets, and the third internal link is designed for displaying the pressure value of pilot chamber. Linking diagram for model is shown in figure B-3.

Model Input:	Pressure in bar	EFFORT
Model Output:	Flow rate in l/s	FLOW

A typical sub-circuit containing model PCA1 together with its block diagram is shown in figure B-4.

The link ordering during program generation must correspond to the port ordering shown in figure B-3. The only acceptable input to the program generator for the sub-circuit shown in figure B-4 is:

PCA101 06 07 08 09 10

PIA5 - A PIPE INSTANTANEOUS MODEL WITH ITERATIVE PROCEDURE

1. Introduction.

PIA5 simulates the instantaneous behaviour of a frictionless pipe. Fluid compressibility, fluid friction and fluid inertia effects are not taken into account. Cavitation is allowed when the working pressure is below the saturation pressure. An iterative procedure is adopted to compute the pipe pressure instantaneously.

The model can only be used for simulations using Runge-Kutta integration subroutine.

2. Model Assumptions.

The following assumptions have been made in the model:

(i) The pressure drop due to pipe friction is assumed to be insignificant and are therefore not accounted for in this model.

(ii) Transient pressure variations due to fluid inertia (momentum effects) are assumed to be insignificant and are therefore not accounted for in this model.

(iii) The algebraic sum of the flow rates to and from the model is zero.

(iv) The change of pressure occurs instantaneously.

3. Nomenclature

P - Pipe pressure in bar

P_{max} - Maximum operating pressure in bar

Q_i - Input flow rate of ith component linked to the pipe in L/s

Q_{max} - Maximum flow rate through pipe in l/s

4. User defined parameters

(i) - initial pipe pressure in bar.

(ii) - maximum flow rate through pipe in l/s

(iii) - maximum operating pressure in bar

5. Model equations.

The pressure value in the model is computed instantaneously by iteration. Following tolerance proportional functions are designed to calculate the pressure and flow error tolerances of iteration in terms of maximum pressure and flow values supplied by the user.

a) Pressure tolerance

$$\Delta p = \frac{P_{max}}{1000} \quad \text{bar}$$

b) Flow tolerance

$$\Delta f = \frac{Q_{max}}{10000} \quad m^3/s$$

a) Set of Pseudo-state Variable. In order to overcome linking difficulty caused by the implicit relationship between non-memory algebraic models, the pressure P in the model must be set to be a pseudo-state variable whose derivative is zero. i.e.

$$\frac{dP}{dt} = 0$$

Since P is a state variable, the initial condition of pressure is specified by the user, or by default as zero. However the pressure P at $t > 0$ is computed instantaneously by an iterative procedure and will not be changed by the integrator.

b) Iteration to compute pipe pressure. Suppose an initial iterative interval $[P_{UP}, P_{LOW}]$ has been obtained according to an automatic procedure shown in section 7, the iterative process commences by estimating a value for pipe pressure using the equation:

$$P^{(k)} = \frac{P_H + P_L}{2} \qquad \ldots\ldots (B.12)$$

where $P_H = P_{UP}$ and $P_L = P_{LOW}$

where k is the number of iteration (initially k=1).

and then a real pressure value P for pipe can be found in the interval $[P_H, P_L]$ by following stages:

i) The pressure $P^{(k)}$ obtained from above equation is used to calculate the flow error (algebraic sum of input and output flow rates of the pipe) by calling adjacent models.

ii) flow and pressure errors are estimated by:

a) flow error: $\Delta Q = \sum Q$

b) pressure error: $\Delta P = P^{(k)} - P^{(k-1)}$

iii) Halving iterative interval for the estimation of value of P, if necessary.

a) If the flow error between the flows is within a set tolerance Δf, i.e.

$$| \Delta Q | \leqslant \Delta f$$

the pressure value chosen $P^{(k)}$ is considered acceptable, i.e.

$$P = P^{(k)}$$

and the iteration for pressure P at this time point is finished.

b) $| \Delta Q | > \Delta f$ but $| \Delta P | \leqslant \Delta p$. It shows that although the flow error is outside the flow tolerance Δf the pressure chosen has been near the real value for P, i.e. pressure error obtained has been within a set tolerance Δp. In this case the estimated pressure value $P^{(k)}$ may be considered acceptable, i.e.

$$P = P^{(k)}$$

and then iterative process is finished.

c) Otherwise the calculations are repeated using a new halved iterative interval $[P_L, P_H]$, when

$$P_L = P^{(k)} \text{ and } P_H = P_H \quad \text{for } \Delta Q > 0 \qquad \ldots\ldots \text{(B.13)}$$

or

$$P_L = P_L \text{ and } P_H = P^{(k)} \quad \text{for } \Delta Q < 0 \qquad \ldots\ldots \text{(B.14)}$$

The iteration process continues until an acceptable value for P is obtained.

Cavitation region. In this model cavitation is allowed when the working

pressure is lower than the saturation pressure and the minimum working pressure is -0.999 bar, i.e.

$P < P_{sat}$ entering cavitation

if $P < -1$ bar, $P = -0.999$ bar

6. Model Linking

The model has 8 external links which receive flow and supply pressure. The external links may be specified in any order. The link diagram for model PIA5 is shown in figure B-5.

Model inputs:

Flow (L/s) (available on up to 8 ports). FLOW

Model outputs:

Pressure in bar (available on up to 7 ports). EFFORT

A typical sub-circuit containing model PIA5, together with its block diagram is shown in figure B-6.

The link ordering during program generation must correspond to the port ordering shown in figure B-5. The only acceptable input to the program generator for the sub-circuit shown in figure B-6 is:

PIA501 08 07 09

7. Automatic procedure of determining initial iterative interval.

See paragraphs 4.24 to 4.29.

PIA6 - PIPE CONNECTED TO ACTUATOR ALLOWING TO EXTREMELY SMALL VOLUMES SHOULD THEY OCCUR AT OR NEAR THE END OF THE ACTUATOR STROKE

1. Introduction.

This model calculates the pressure in a pipe for which friction effects are neglected and it is a model of a pipe connected to an actuator, taking into account the variable volume which occurs as the actuator moves. The effects of elasticity of pipe walls and compressibility of the fluid are taken into account, except in the case where the combined pipe/actuator volume is extremely small. In such cases, the model is arranged so that it will compute the pressure changes as though they were instantaneous. This is necessary to prevent an excessive computing time which would occur with very small volumes. The limiting volume is specified by the user.

If the fluid volume V_{IN} of the actuator is greater than the critical volume V_c, cavitation is allowed for by assuming an air release mechanism. At low operating pressures, the model calculates the proportion by volume of the air released from solution and the effect of this free air on the effective bulk modulus of the fluid. The model allows the specification of the pressure at which the fluid is saturated with air, that is the pressure below which air release will occur and will give a reasonably accurate representation of cavitation in most cases.

At the volume V_{IN} less than the critical volume an iterative procedure is adopted which assumes that the pressure instantaneously attains the value imposed on it by the flows and pressures received from other models. Cavitation is accounted for when the operating pressure of a pipe is below the saturation pressure.

Fluid friction and fluid inertia effects are not taken into account.

The model receives fluid flows as input and supplies pressure to adjoining models.

The model can only be used for simulation using Runge-Kutta integration subroutine.

2. Model assumptions.

The following assumptions have been made in the model:

(i) The pipe volume is very small comparing with that of actuator.

(ii) The pipe flow rates (including predicted flow rates) provided as model inputs will be calculated in other models.

(iii) Pressure drops due to pipe friction are assumed to be insignificant and are therefore not accounted for in this model.

(iv) Transient pressure variations due to fluid inertia (momentum effects) are assumed to be insignificant and are therefore not accounted for in this model.

For the dynamic integration region:

(i) The air released from solution will be expanded polytropically with a constant index of 1.4.

(ii) The pipe fluid flow rates provided as model inputs will be valid during cavitation. These flows will be calculated in other models on the basis of single phase liquid flow. In fact two phase flow will exist during cavitation and the model will be in error in the event of flow from a cavitating pipe into another cavitating pipe through a control valve.

(iii) All flow from components connected to a pipe contains the same proportions of air and fluid that exist in the pipe.

(iv) Cavitation will cease immediately when flow sufficient to compress the free air to the saturation condition, has re-entered the pipe. This should be reasonably accurate for conditions where fluid is continually being replenished as in the case of a hydraulic transmission system.

(v) The release of fluid vapour will only become significant under extreme cavitation conditions and can be ignored.

For the instantaneous iterative region:

(i) Pressure changes occur immediately.

3. Nomenclature

a' - Proportion (measured by volume) of free air at working pressure
a - Proportion (measured by volume at STP) of free air at working pressure
β_a - Bulk modulus of air in bar
β_e - Effective bulk modulus of fluid or fluid-air combination in bar
β_f - Bulk modulus of fluid in bar
d_0 - Proportion (measured by volume) of air dissolved in the fluid at STP
$f(P)$ - Function relating to the fluid compressibility
d_{sat} - Proportion (measured by volume) of air dissolved in
at saturation pressure
h - Cubic smoothing interval in bar
P - Pipe pressure in bar
P_{sat} - Pressure at which fluid is saturated with air in bar
P_{max} - Maximum operating pressure in bar
Q - Net pipe flow rate in L/s
Q'_a - Rate of air release measured at STP in L/s
Q_a - Rate of air release measured at working pressure in L/s
Q_i - Input flowrate of ith component linked to the pipe in L/s
Q_{max} - Maximum flow rate through pipe in l/s
n - Polytropic index
V_a - Volume of air in pipe control volume in L
V_c - Critical volume in L
V_{IN} - Volume of fluid in actuator piston side in L
V_p - Volume of pipe in L
V_t - Total volume in L

4. User defined parameters

(i) - pipe internal diameter in mm.
(ii) - pipe length in m.
(iii) - Either effective bulk modulus in bar or pipe wall thickness and internal diameter in mm and pipe material (in order to determine the compliance of the pipe). The pipe materials at present available are steel, tungum, cupro-nickel and flexible hose.
(iv) - pressure at which oil is saturated with air in bar.
(v) - proportion of air by volume, dissolved in oil at standard temperature and pressure.
(vi) - initial pipe pressure in bar.
(vii) - maximum flow rate through pipe in l/s
(iii) - maximum operating pressure in bar

5. Model equations.

According to the comparison between the actuator piston side chamber volume V_{IN} and a setting critical volume V_c, two operating regions of model are defined:

$V_{IN} > V_c$ - the dynamic region in which the integration is used.

and $V_{IN} \leqslant V_c$ - the instantaneous region in which the iteration is used.

The critical volume V_c is defined as follows:

$$V_c = 10\% \, V_{A\,max}$$

or $$X_A = 10\% \, X_{A\,max}$$

where X_A - displacement of actuator piston
$X_{A\,max}$ - maximum displacement of actuator piston
$V_{A\,max}$ - maximum volume of actuator piston chamber

Dynamic operating region.

If the piston side chamber volume V_{IN} is larger than the setting critical volume V_c (e.g. $V_c = 10\% V_{Amax}$), the fluid compressibility is considered, and hence the dynamic behaviour of this pipe is described by the following flow continuity equation:

$$\frac{dP}{dt} = \frac{\beta_e}{V_p + V_{IN}} \sum Q_i$$

where the pressure P is a state variable and will be supplied by the integrator. β_e is an effective bulk modulus and is computed by

Case 1: Operation above the pressure at which the fluid is saturated with air, P_{sat}

$$\beta_e = \beta_f \qquad \ldots\ldots (B.15)$$

Case 2: Operation below P_{sat}

$$\beta_e = \frac{1}{\frac{1}{\beta_f} + \frac{d_0 (P_{sat} - P)}{n (P+1)^2}} \qquad \ldots\ldots (B.16)$$

Therefore in this dynamic region, the pressure transient behaviour is dependent with the fluid compressibility, and the mathematical model allows the cavitation by assuming an air release mechanism in above β_e equation (Ref.26).

Instantaneous region.

If the piston side chamber volume V_{IN} is equal to or less than a setting critical volume V_c, the fluid compressibility is assumed to be negligible, and hence an iterative procedure is adopted to compute pressure instantaneously by means of a set of flow and pressure tolerances specified below.

a) Pressure tolerance

$$\Delta p = \frac{P_{max}}{1000} \quad \text{bar}$$

b) Flow tolerance

$$\Delta f = \frac{Q_{max}}{10000} \quad m^3/s$$

Set of Pseudo-state Variable. In this region, in order to overcome possible linking difficulty caused by the implicit relationship between non-memory algebraic models, the pressure P in the model must be set to be a pseudo-state variable whose derivative is zero. i.e.

$$\frac{dP}{dt} = 0$$

Iteration to compute pipe pressure. Suppose an initial iterative interval $[P_{UP}, P_{LOW}]$ has been obtained according to an automatic procedure shown in section 7, the iterative process commences by estimating a value for pipe pressure using the equation:

$$P^{(k)} = \frac{P_H + P_L}{2} \qquad \ldots\ldots (B.17)$$

where $P_H = P_{UP}$ and $P_L = P_{LOW}$

where k is the number of iteration (initially k=1).

and then a real pressure value P for pipe can be found in the interval $[P_H, P_L]$ by following stages:

i) The pressure $P^{(k)}$ obtained from above equation is used to calculate the flow error (algebraic sum of input and output flow rates of the pipe) by calling adjacent models.

ii) flow and pressure errors are estimated by:

a) flow error: $\Delta Q = \sum Q$

b) pressure error: $\Delta P = P^{(k)} - P^{(k-1)}$

iii) Halving iterative interval for the estimation of value of P, if necessary.

a) If the flow error between the flows is within a set tolerance Δf, i.e.

$$| \Delta Q | \leqslant \Delta f$$

the pressure value chosen $P^{(k)}$ is considered acceptable, i.e.

$$P = P^{(k)}$$

and the iteration for pressure P at this time point is finished.

b) $| \Delta Q | > \Delta f$ but $| \Delta P | \leqslant \Delta p$. It shows that although the flow error is outside the flow tolerance Δf the pressure chosen has been near the real value for P, i.e. pressure error obtained has been within a set tolerance Δp. In this case the estimated pressure value $P^{(k)}$ may be considered acceptable, i.e.

$$P = P^{(k)}$$

and then iterative process is finished.

c) Otherwise the calculations are repeated using a new halved iterative interval $[P_L, P_H]$, when

$$P_L = P^{(k)} \text{ and } \quad P_H = P_H \quad \text{for } \Delta Q > 0 \qquad \ldots\ldots (B.18)$$

or

$$P_L = P_L \text{ and } \quad P_H = P^{(k)} \quad \text{for } \Delta Q < 0 \qquad \ldots\ldots (B.19)$$

The iteration process continues until an acceptable value for P is obtained.

Cavitation region. In this model cavitation is allowed when the working pressure is lower than the saturation pressure and the minimum working pressure is -0.999 bar, i.e.

$P < P_{sat}$ entering cavitation

if $P < -1$ bar, $P = -0.999$ bar

6. Model Linking.

The model accepts flow rate as an output requirement on from 1 to 8 links and provides pressure as an output on each external link. The link diagram for model PIA6 is shown in figure B-7.

Model inputs:

Flow (L/s) (available on up to 8 ports). FLOW

Model outputs:

Pressure in bar (available on up to 7 ports). EFFORT

A typical sub-circuit containing model PIA6, together with its block diagram is shown in figure B-8.

The link ordering during program generation must correspond to the port ordering shown in figure B-7. The only acceptable input to the program generator for the sub-circuit shown in figure B-8 is:

PIA601 07 08 01

7. Automatic procedure of determining initial iterative interval.

See paragraphs 4.24 to 4.29.

SHA1 - A SHAFT MODEL WITH ITERATIVE PROCEDURE

1. Introduction.

This model simulates the behaviour of a lossless circular shaft using iterative technique and integrator. The flexibility, torque loss and inertia (or mass) effects of the shaft are neglected. The shaft torque and angular acceleration values are computed instantaneously by an iterative procedure. The model requires the solution of a first order differential equation to represent the angular acceleration of the shaft. It therefore computes one state variable which is angular speed ω.

The model can only be used for simulations using Runge-Kutta, Runge-Kutta-Merson, or other single step integration subroutines.

2. Model Assumptions.

The model assumes that the circular shaft is considered to be infinitely rigid and the torque dependent mechanical friction loss is ignored, and hence the angular acceleration and speed acting on the both sides of the shaft may be considered identical. Transient torque variations due to shaft inertia are assumed to be insignificant and are therefore not accounted for in this model.

3. Nomenclature

J_D - Inertia of driving component (eg. Engine or hydraulic motor) in kgm^2
J_L - Load inertia in kgm^2
T_D - Driving torque in Nm
T_L - Load torque in Nm
T_{max} - Maximum operating torque acting on shaft in Nm
Ts - Average torque acting on shaft in Nm

Ts_1, Ts_2- Estimated torque value acting on the both sides of shaft in Nm

$\dot{\omega}$ - Average angular acceleration on shaft in $1/s^2$

$\omega_{\dot{1}}$,$\omega_{\dot{2}}$ - Estimated angular acceleration on both sides of shaft in $1/s^2$

$\dot{\omega}_{max}$ - Maximum angular acceleration on shaft in $1/s^2$

4. User defined parameters

(i) - initial shaft angular speed in 1/s

(ii) - maximum operating torque acting on shaft in Nm

(iii) - maximum angular acceleration on shaft in 1/s

(iv) - inertia of driving component (eg.engine or motor) in kgm^2

(v) - load inertia in kgm^2

5. Model equations.

Iteration to compute shaft torque and angular acceleration. Models containing instantaneous equations could be implemented using the iterative technique. In this model the shaft torque and acceleration values are evaluated respectively by using following equations:

i) Assume $\dot{\omega}_{n-1}$, T_D and T_L are known. Two estimates of torque acting on the both sides of shaft are given by:

$$Ts_1 = T_D - J_D \dot{\omega}_{n-1}$$

$$Ts_2 = T_L + J_L \dot{\omega}_{n-1}$$

ii) We can average these to get

$$T_s = \frac{Ts_1 + Ts_2}{2} \qquad \ldots\ldots (B.20)$$

We can then obtain two values for acceleration $\dot{\omega}$. i.e. $\dot{\omega}_1$ and $\dot{\omega}_2$.

$$\dot{\omega}_1 = \frac{T_D - T_s}{J_D}$$

$$\dot{\omega}_2 = \frac{T_s - T_L}{J_L}$$

And an average can be obtained by employing the equation:

$$\dot{\omega} = \frac{\dot{\omega}_1 + \dot{\omega}_2}{2} \qquad \ldots\ldots (B.21)$$

The process can be iterated until preset tolerances for T_s and $\dot{\omega}$ are reached. The T_s and $\dot{\omega}$ are what we want to find.

Transfer of inertias to shaft model. Above iteration calculation in shaft model requires inertia values J_D and J_L from adjacent driving and load components. These values may be variable with respect to time and will be supplied through two signal links on the model. Figure 5.4 shows the free-body diagram showing torques acting on shaft and adjacent models and figure 5.5 shows a transfer of inertias to shaft model.

Determination of iterative tolerances. Following tolerance proportional functions are designed to calculate the torque and angular acceleration tolerances of iteration in terms of maximum torque T_{max} and angular acceleration $\dot{\omega}_{max}$ supplied by the user.

i) Torque tolerance

$$\Delta T = \frac{T_{max}}{10000}$$

ii) Angular acceleration tolerance

$$\Delta\dot{\omega} = \frac{\dot{\omega}_{max}}{100000}$$

Obtain Shaft Angular Speed Value. Once a value for shaft angular acceleration $\dot{\omega}$ is obtained, the value for shaft angular speed ω can be obtained by integration.

6. Model Linking

The model has 2 external links and 2 signal links. The external links connected to ports 1 and 2 receive torque and supply angular speed. The signals attached to the model are inputs transmitting the inertia from adjacent models. The linking diagram for model SHA1 is shown in figure B-9.

Model inputs:

Torque in NM (available on up to 2 ports) EFFORT

Model outputs:

Angular speed (1/S) (available on up to 2 ports). FLOW

A typical sub-circuit containing model SHA1, together with its block diagram is shown in figure B-10.

The link ordering during program generation must correspond to the port ordering shown in figure B-9. The only acceptable input to the program generator for the sub-circuit shown in figure B-10 is:

SHA101 10 11

SHA2 - A SHAFT-GEARBOX UNIT MODEL WITH AN ITERATIVE PROCEDURE

1. Introduction.

This model comprises a gearbox and its input and output shafts and is used to simulate the behaviour of input and output shafts of the gearbox by iterative technique and integrator. A schematic of a shaft-gearbox unit is shown in figure 5.10. The flexibility, torque loss and inertia (or mass) effects of the shaft are neglected. The torque and angular acceleration values for the gear box input shaft are computed instantaneously by an iterative procedure. The model requires the solution of a first order differential equation to represent the angular acceleration of the input shaft, and hence it computes one state variable which is angular speed ω of input shaft. Afterwards the angular speed of the output shaft is obtained by means of the gear box ratio value.

The model can only be used for simulations using Runge-Kutta, Kutta-Merson or other single step integration subroutines.

2. Model Assumptions.

The model assumes that the circular shaft is considered to be infinitely rigid and the torque dependent mechanical friction loss is ignored, and hence the angular acceleration and speed acting on the both sides of one shaft may be considered identical. Transient torque variations due to shaft inertia are assumed to be insignificant and are therefore not accounted for in this model. The transmission efficiency of the gear box is 100%.

3. Nomenclature

G - Gear box ratio
J_D - Inertia of driving component (eg. Engine or hydraulic motor) in kgM^2
J_L - Load inertia of output shaft in kgm^2
J_{G1} - First gear wheel inertia in kgm^2
J_{G2} - Second gear wheel inertia in kgm^2
J_{Leq} - Equivalent value for the load inertia of input shaft in kgm^2
T_D - Driving torque in Nm
T_L - Load torque of output shaft in Nm
T_{Leq} - Equivalent value for the load torque of input shaft in Nm
T_{max} - Maximum operating torque acting on shaft in Nm
T_s - Average torque acting on the input shaft in Nm
T_{s1}, Ts_2- Estimated torque acting on the both sides of input shaft in Nm
ω_i, ω_2 - Estimated angular acceleration of both sides of input shaft in $1/s^2$
$\dot{\omega}$ - Average angular acceleration of input shaft in $1/s^2$
$\dot{\omega}_L$ - Angular acceleration of output shaft in $1/s^2$
$\dot{\omega}_{max}$ - Maximum angular acceleration of shaft in $1/s^2$
ω - Angular speed of input shaft in $1/s^2$
ω_L - Angular speed of output shaft in $1/s^2$

4. User defined parameters

(i) - initial angular speed of input shaft in 1/s
(ii) - maximum operating torque acting on shaft in Nm
(iii) - maximum angular acceleration of shaft in 1/s
(iv) - inertia of driving component in kgm^2
(v) - load inertia in kgm^2
(vi) - gear box ratio

5. Model equations.

This model supplies two different angular speeds to adjoining models due to the gear box ratio. One is the value of the input shaft and another is of the output shaft. The

torque T_s and angular acceleration ω of the input shaft are computed instantaneously by iteration, and the values for the output shaft may be computed by means of the variable relationships between input and output shafts of a gear box which are given below.

Variable relationships between input and output shafts of a gear box

i) angular acceleration:

$$\dot{\omega} = G\ \dot{\omega}_L$$

ii) angular speed:

$$\omega = G\ \omega_L$$

iii) shaft torque:

$$T_s = \frac{T_{sL}}{G}$$

iv) Equivalent value for the load inertia of input shaft:

$$J_{Leq} = J_{G1} + \frac{J_L + J_{G2}}{G^2}$$

Iterative Procedure for Torque and Acceleration Values of gearbox input shaft. Supposing the conditions at time t_{n-1} are known, for instance, ω_{n-1} and $\dot{\omega}_{n-1}$, the torque and angular acceleration values of the gearbox input shaft can be computed instantaneously by the iterative procedure given below.

i) Estimate the torque acting on the shaft by using the driving torque T_D and load torque T_L. Two estimates of torque acting on the two sides of input shaft are given by

$$Ts_1 = T_D - J_D\ \dot{\omega}_{n-1}$$

$$Ts_2 = T_{Leq} + J_{Leq}\ \dot{\omega}_{n-1}$$

We can get an average torque acting on the shaft

$$T_s = \frac{Ts_1 + Ts_2}{2} \qquad \ldots\ldots (B.22)$$

ii) We can then obtain two values for acceleration $\dot{\omega}$, i.e. $\dot{\omega}_1$ and $\dot{\omega}_2$.

$$\dot{\omega}_1 = \frac{T_D - T_s}{J_D}$$

$$\dot{\omega}_2 = \frac{T_s - T_{Leq}}{J_{Leq}}$$

And an average can be obtained by employing the equation:

$$\dot{\omega} = \frac{\dot{\omega}_1 + \dot{\omega}_2}{2} \qquad \ldots\ldots (B.23)$$

The process can be iterated until preset tolerances for T_s and $\dot{\omega}$ are reached.

Determination of iterative tolerances. Following tolerance proportional functions are designed to calculate the torque and angular acceleration tolerances of iteration in terms of maximum torque *Tsmax* and angular acceleration $\dot{\omega}_{max}$ supplied by the user.

i) Torque tolerance:

$$\Delta T = \frac{T_{max}}{10000}$$

ii) Angular acceleration tolerance:

$$\Delta\dot{\omega} = \frac{\Delta\omega_{max}}{100000}$$

Solution for Angular Speed of Gear Box Input Shaft. Once a value for shaft angular acceleration $\dot{\omega}$ is obtained, the value for shaft angular speed ω can be supplied by integration.

Solution for Angular Speed of Gear Box Output Shaft. According to the variable relationships between the input and output shafts of a gear box, the angular speed of the output shaft can be obtained as follows:

$$\omega_L = \frac{\omega}{G}$$

Set of Pseudo-state Variable. In order to avoid the linking problem between shaft-gearbox unit model and other models, the variable ω_L in this model must be set as a pseudo-state variable whose derivative is zero. i.e.

$$\frac{d\,\omega_L}{dt} = 0$$

In this case, the value for ω_L still depends on the ω and the gear ratio, but will not be affected by the integrator.

6. Model Linking

The model has 2 external links and 2 signals. The external link connected to port 1 receives torque and supplies angular speed of input shaft of gear box. The external link connected to port 2 receives load torque and supplies angular speed of output shaft of gear box. The signals attached to the model are inputs transmitting the inertia from adjacent models. The linking diagram for model SHA2 is shown in figure B-11.

Model inputs:

Torque in NM (available on up to 2 ports) EFFORT

Model outputs:

Angular speed (1/S) (available on up to 2 ports). FLOW

A typical sub-circuit containing model SHA1, together with its block diagram is shown in figure B-12.

The link ordering during program generation must correspond to the port ordering shown in figure B-11. The only acceptable input to the program generator for the sub-circuit shown in figure B-12 is:

SHA201 10 11

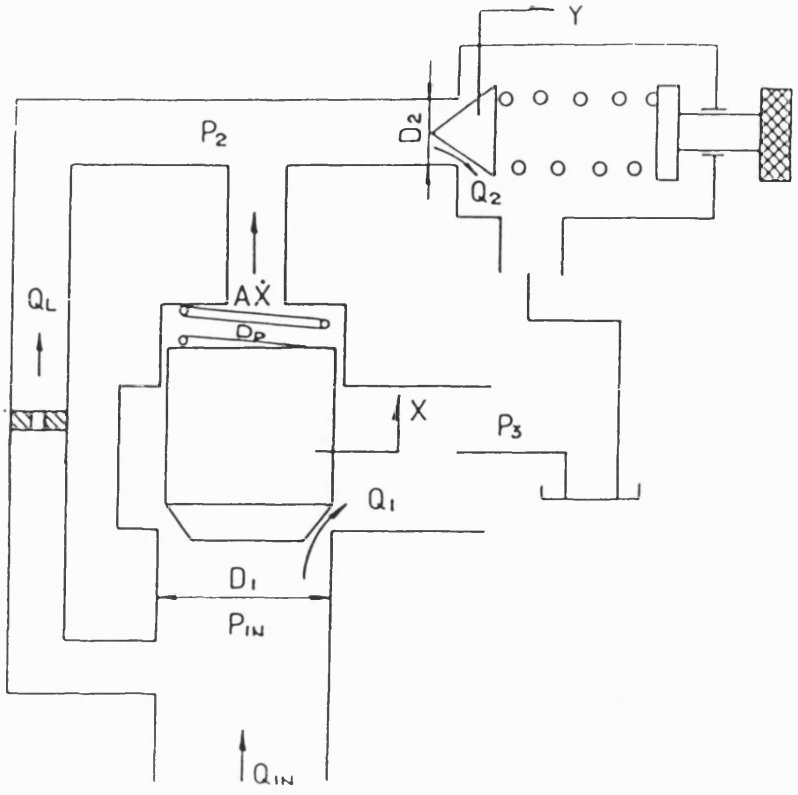


FIG. B-1 Schematic of a Two Stage Relief Valve

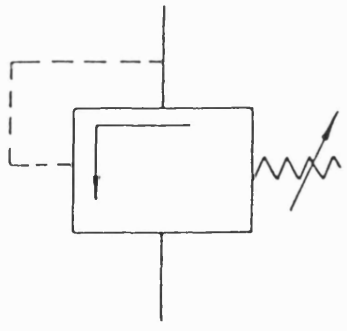


FIGURE B-2 DESIGN SYMBOL FOR A TWO STAGE RELIEF VALVE

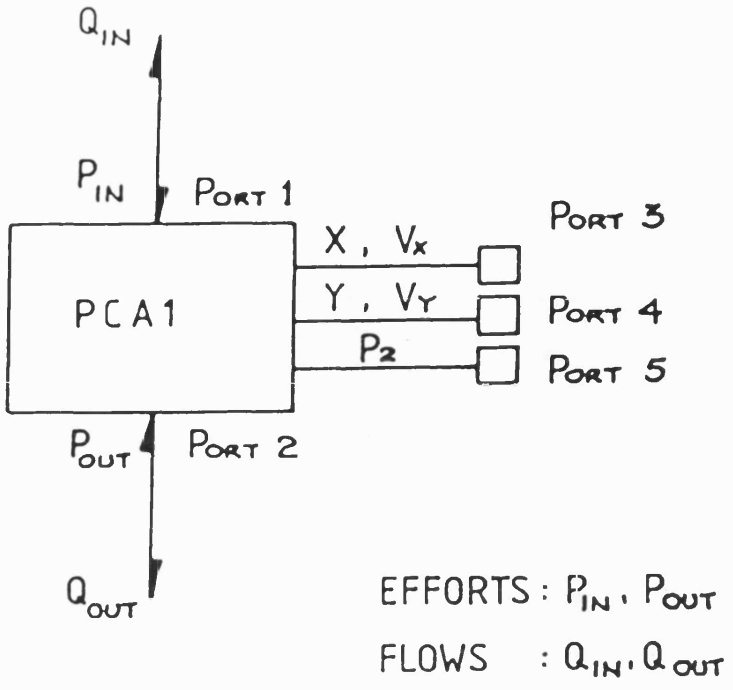


FIGURE B-3 LINK DIAGRAM FOR TWO STAGE RELIEF VALVE MODEL PCA1

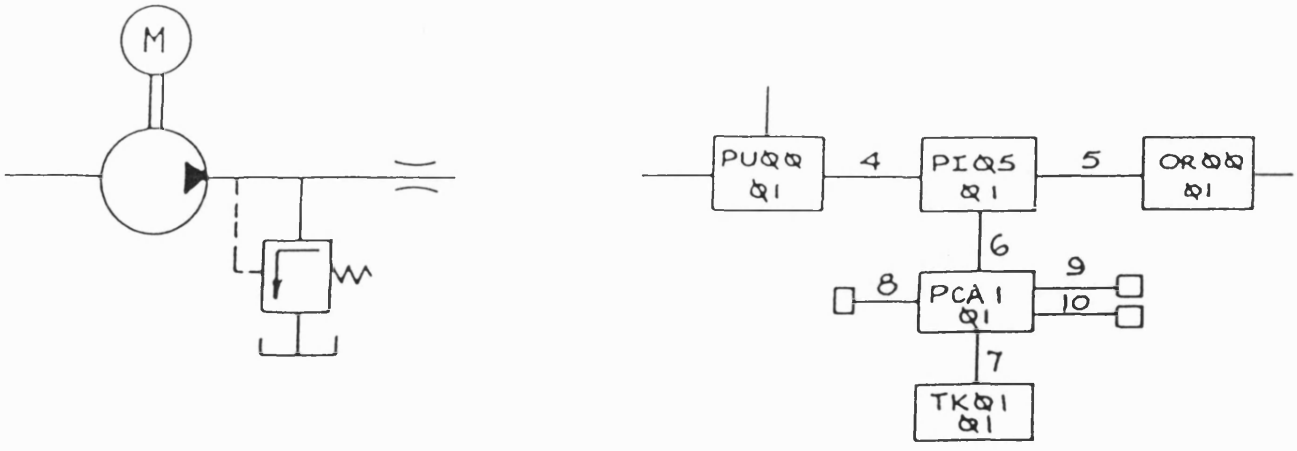


FIGURE B-4 SUB-CIRCUIT BLOCK DIAGRAM FOR SIMULATION CORRESPONDING TO MODEL PCA1

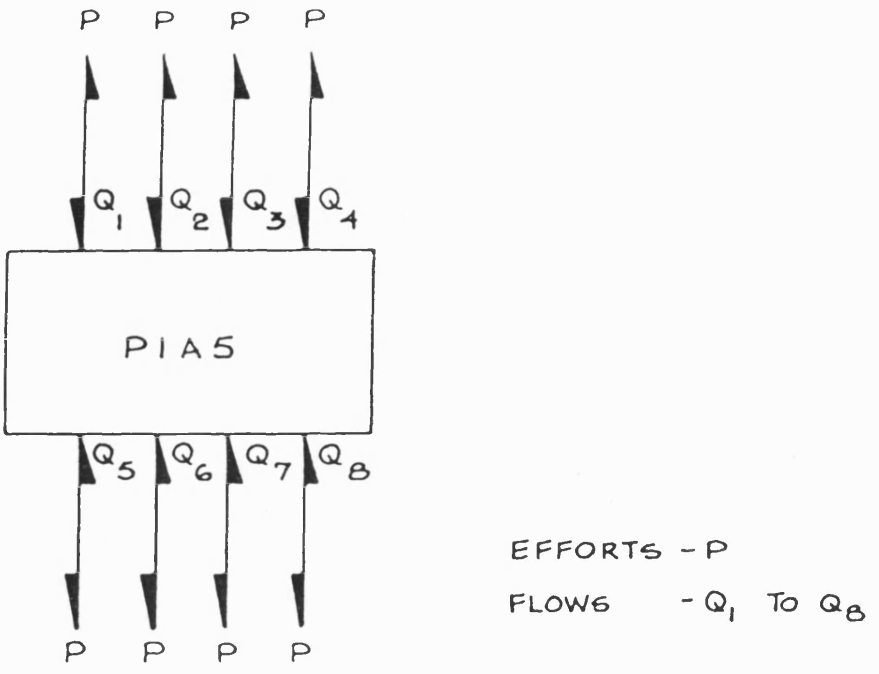


FIGURE B-5 LINK DIAGRAM FOR PIA5 - AN ITERATIVE MODEL OF A PIPE

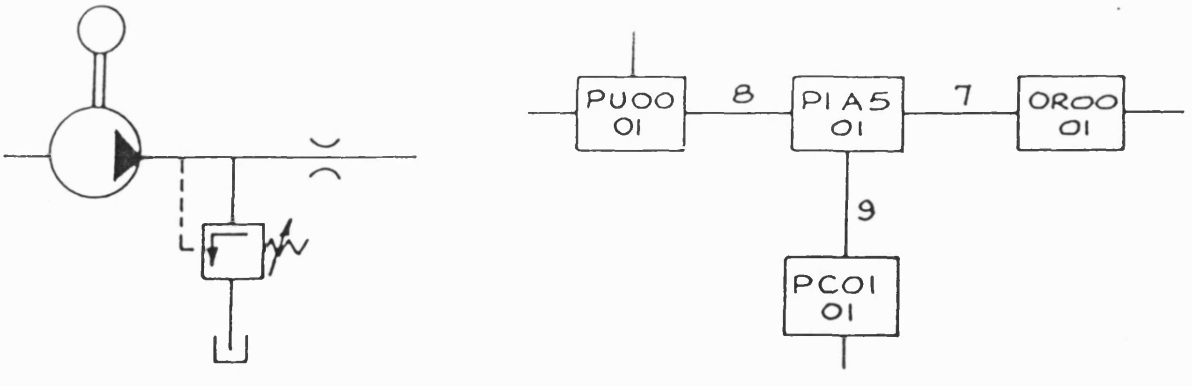
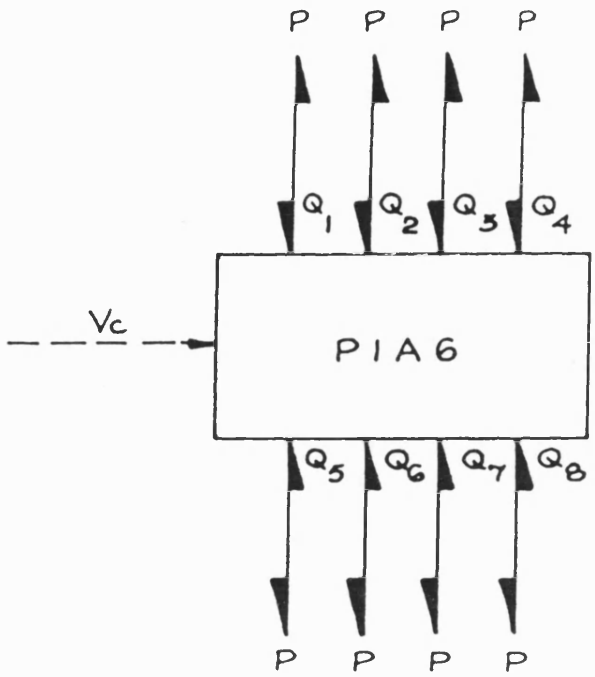


FIGURE B-6 SUB-CIRCUIT AND BLOCK DIAGRAM FOR SIMULATION USING MODEL PIA5



EFFORTS : P

FLOWS : Q_1 TO Q_8

FIGURE B-7 LINK DIAGRAM FOR PIA6

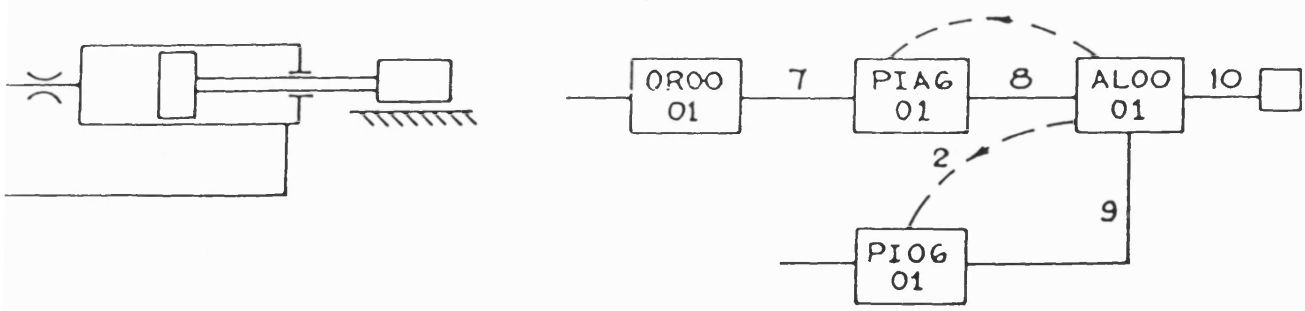


FIGURE B-8 SUB-CIRCUIT AND BLOCK DIAGRAM FOR SIMULATION USING MODEL PIA6

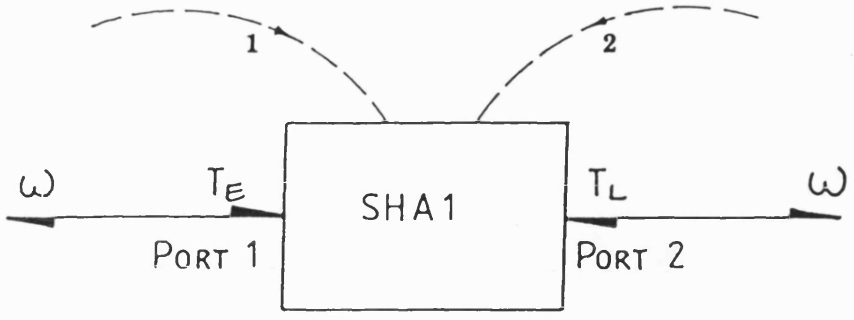


FIGURE B-9 LINK DIAGRAM FOR SHAFT MODEL SHA1

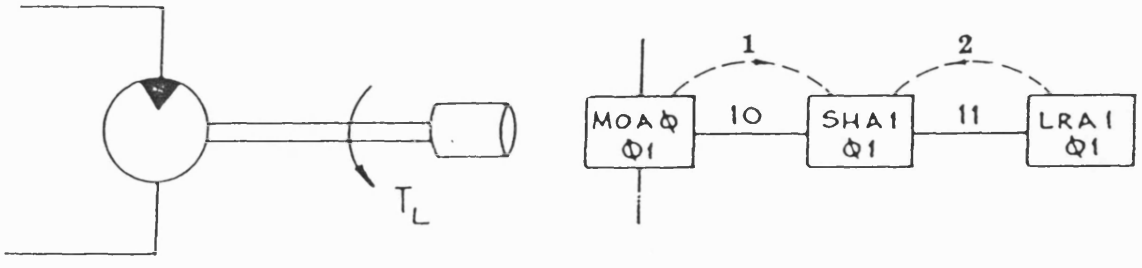
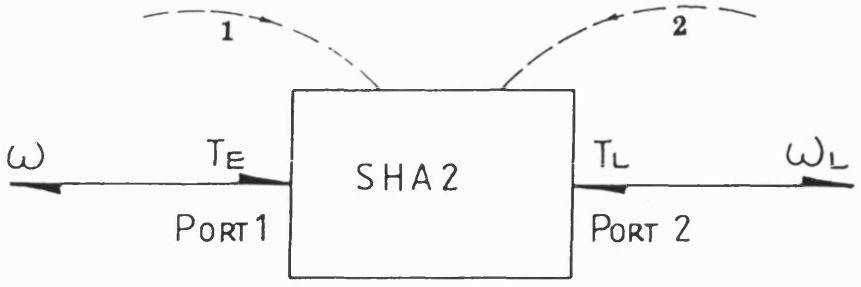


FIGURE B-10 SUB CIRCUIT AND BLOCK DIAGRAM FOR SIMULATION USING MODEL SHA1



EFFORTS : T_E , T_L

FLOWS : ω , ω_L

FIGURE B-11 LINK DIAGRAM FOR SHA2
A SHAFT MODEL WITH A GEAR BOX

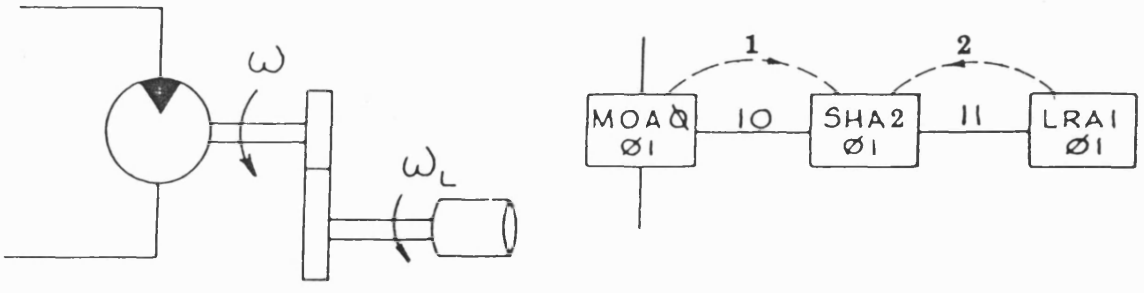


FIGURE B-12 SUB CIRCUIT AND BLOCK DIAGRAM FOR
SIMULATION USING MODEL SHA2

APPENDIX C

DOCUMENTATION OF COMPUTER MODELS

USING SINGLE STEP INTEGRATION METHODS

CONTENTS

N.B. These models are available in the HASP model library.

ALR0 - LINEAR ACTUATOR WITH LOAD MODEL

1. Introduction

ALR0 models the behaviour of a double acting linear actuator and load combination. The model takes into account viscous, windage and stick/slip friction and includes a spring force. A gravitational force is included to account for the inclination of the actuator rod to the horizontal. Piston leakage is also included in the model. The model receives piston and rod end pressures from adjoining models and computes the corresponding flows. Piston and rod end volumes are transmitted to adjoining models. The actuator velocity and displacement are available for display. A schematic of the component is shown in figure C-1 [7].

The model can only be used for simulations using Runge-Kutta or Kutta-Merson integration subroutines.

2. Assumptions

(i) The model assumes no compliance exists between the actuator piston and load i.e. actuator and load displacements are equal.

(ii) The stiction level is assumed to be independent of the time the actuator is at rest.

3. Nomenclature

A_1 - inlet piston cross sectional area in cm^2
A_2 - outlet piston cross sectional area in cm^2
E - modulus of elasticity in N/m
f - coefficient of viscous friction in N/(m/s)

f_w - windage coefficient in $N/(m/s)^2$
F_{ap} - applied force in N
F_s - stiction force in N
F_c - Coulomb friction force in N
k - spring stiffness in N/m
k_p - piston internal leakage coefficient in (L/s)/bar
m - mass of piston and load in kg
P_1 - inlet pressure in bar
P_2 - outlet pressure in bar
x - piston displacement in m
x_1 - actuator stroke in m
v - piston velocity in m/s
V_1 - volume of actuator at inlet in L
V_2 - volume of actuator at outlet in L
θ - angle of inclination of actuator rod to horizontal in degrees

4. User defined parameters

(i) Actuator diameter (cm)
(ii) Rod diameter (cm)
(iii) Total mass of actuator and load (Kg)
(iv) Stiction (N)
(v) Coulomb friction (N)
(vi) Viscous friction coefficient (N/(m/s))
(vii) Windage loss coefficient $N/(m/s)^2$
(viii) Angle of actuator inclination (degrees)
(ix) Actuator stroke (m)
(x) Initial displacement of actuator (m)
(xi) Initial velocity of actuator (m/s)
(xii) Spring stiffness (N/m)
(xiii) Piston leakage coefficient ((L/s)/bar)

5. Model equations

Model ALR0 has two important features which require explanation. These are the representation of stick/slip friction and the actuator end stops.

(i) Consideration of stick slip friction.

If the linear actuator is moving, a constant frictional force opposes its motion. This is termed Coulomb friction. If the actuator is at rest and the net applied force is less than the friction (which increases with increasing applied force), then the net total force on the actuator is zero. In this condition the actuator remains at rest. The maximum (or limiting) value of friction, always greater than or equal to the Coulomb friction, is termed stiction as the actuator is stationary. As soon as the net applied force is sufficient to overcome stiction, the frictional force is assumed to drop instantaneously from the stiction level to the Coulomb friction level. The model outlined assumes an instantaneous change in the friction level when stiction is overcome and a sudden change in direction as the actuator velocity changes sign. In modelling stick/slip friction reliably, a suitable cubic function is used to represent the discontinuity.

The force acting on the actuator is given by the following second order differential equation.

(a) For $|v| > 0$

$$P_1A_1 - P_2A_2 = m\ \frac{dv}{dt} + fv\ + f_w v^2 + mg\ \sin\theta + kx\ + F_c\ sign(v) \quad \ldots\ldots (C.1)$$

where

$$\frac{dv}{dt} = \frac{d^2x}{dt^2}$$

This equation can be rewritten as the following two first order differential equations

$$\frac{dx}{dt} = v \quad \text{(defining the time derivative of piston displacement)}$$

and

$$\frac{dv}{dt} = \frac{1}{m}\left(P_1 A_1 - P_2 A_2 - fv - f_w v^2 - mg\ \sin\theta - kx - F_c\ sign\,(v)\right)$$

defining the time derivative of the piston velocity (acceleration). The velocity independent frictional force has magnitude F_c in this case and opposes the velocity.

(b) For $v = 0$ and $|F_{ap}| > |F_s|$

where the net applied force F_{ap} is given by:

$$F_{ap} = P_1 A_1 - P_2 A_2 - mg\ \sin\theta - kx$$

In this case, the piston velocity is zero but the net applied force is greater than the limiting stiction value F_s. The velocity independent frictional term is assumed to be the Coulomb friction level F_c. The piston acceleration is given by:

$$\frac{dv}{dt} = \frac{1}{m}\left(P_1 A_1 - P_2 A_2 - mg\ \sin\theta - kx - F_c\ sign\,(F_{ap})\right) \qquad \ldots\ldots \text{(C.2)}$$

(c) For $v = 0$ and $|F_{ap}| < |F_s$

In this case the piston is assumed to be stationary and

$$\frac{dv}{dt} = 0 \qquad \ldots\ldots (C.3)$$

Note that if the piston is stationary and the applied force becomes greater than the limiting stiction value F_s , the velocity independent friction immediately drops to the Coulomb friction level F_c (from the stiction level F_s) as shown in figure C-2. A further point is that in this model the stiction value is assumed constant. In reality it is time dependent, tending to increase the longer the piston remains at rest.

(ii) Consideration of end stops.

The piston motion of the actuator between the end stops ($0 < X < X_{max}$) is described by the equation (C.2). However this equation is not suitable for the actuator because the real end-stop reaction forces acting on the actuator piston are not considered. If there is no any modification for the equation of motion, in the low end-stop the displacement of the actuator might be less than zero and in the high end-stop might be greater than X_{max}. That is not true because in practical working conditions the piston must be at rest when it hits the end stop. A better method for handlling this kind of the difficulty is to use the "IF STATEMENTS" technique in the component model subroutine so that the displacement "overshoot" or "undershoot" existed in the simulation can be limited, and the velocity of the piston also can be limited to be zero when the actuator hits a stop. The equations for end-stops can be given by:

$$\text{if } X > X_{max} \text{ or } X < 0 \qquad \dot{v} = 0$$

$$\text{if } X < 0 \qquad \text{then } X = 0 \qquad \text{and} \qquad v = 0$$

$$\text{if } X > X_{max} \qquad \text{then } X = X_{max} \qquad \text{and} \qquad v = 0$$

(iii) **Flow rates at inlet and outlet of actuator.**

Inlet (piston end) flow rate.

$$Q_1 = -(A_1 v + k_p (P_1 - P_2)) \quad \ldots\ldots (C.4)$$

Outlet (rod end) flow rate.

$$Q_2 = -(A_2 v + k_p (P_2 - P_1)) \quad \ldots\ldots (C.5)$$

Incremental volumes at inlet and outlet ofactuator.

Incremental volume at inlet (piston end) of pipe

$$\Delta V_1 = A_1 x \quad \ldots\ldots (C.6)$$

Incremental volume at outlet (rod end) of pipe

$$\Delta V_2 = A_2 (x_1 - x) \quad \ldots\ldots (C.7)$$

6. Model linking

The model has 2 external links, 1 internal link and 2 signals. The external link connected to port 1 receives inlet pressure and supplies inlet flow rate. The external link connected to port 2 receives outlet pressure and supplies outlet flow rate. The internal link connected to port 3 displays actuator displacement (effort) and actuator velocity (flow). The first signal attached to the model is an output transmitting the piston volume. The second signal attached to the model is an output transmitting the rod-end volume. The link diagram for model ALR0 is shown in figure C-3.

Model inputs:

P_{in} - Actuator inlet pressure (piston side) in bar
P_{out} - Actuator outlet pressure (rod side) in bar

Model outputs:

Q_{in} - Actuator inlet flow rate in l/s
Q_{out} - Actuator outlet flow rate in l/s
x - Actuator displacement in m
v - Actuator velocity in m/s
V_1 - piston end volume in l
V_2 - rod end volume in l

The link ordering during program generation must correspond to the port ordering shown in figure C-3.

A typical sub-circuit containing model ALR0, together with its block diagram is shown in figure C-4. The only acceptable input to the program generator for the sub-circuit shown in figure C-4 is:

ALR001 07 06 08 01 02

PCR1 - DYNAMIC MODEL OF A TWO STAGE RELIEF VALVE

1. Introduction

PCR1 is a model of a two stage relief valve. It considers the dynamic behaviour of the valve and takes into account the effects of the steady state and transient flow forces which act on the main spool and pilot poppet. The fluid compressibility, fluid friction in the pilot stage chamber are also taken into account. The model receives inlet and outlet pressure as input and supplies flow to adjoining models. A schematic diagram for a two stage relief valve is shown in figure B-1 and its design symbol is shown in figure B-2.

The model can only be used for simulations using Runge-Kutta or Kutta-Merson integration subroutines.

2. Assumptions

(1) the poppet valves have chamfer no their seats.
(2) the compressibility of the oil in the downstream chamber is neglected.
(3) no radial forces exist on either of the poppet valves
lateral forces acting on the valve are ignored.
(4) transient flow forces acting on poppet valves are ignored.
(5) gravity effects are neglected.
(6) flow is irrotational.

3. Nomenclature

t - time
Q - flow rate
P - pressure

P_{cr} - cracking pressure of a two stage relief valve
β - effective bulk modulus of the system
C_d - coefficient of discharge of the orifice
f - viscous coefficient
C_r - amending coefficient
K - spring stiffness
M - mass
D - diameter
d_o - diameter of control orifice
α - angle of poppet valve or main spool
A - area
V - fluid volume
X, Y - displacements of main spool and pilot poppet valve
v_x, v_y - velocities of main spool and pilot poppet valve
ρ - fluid density

Suffixes

1 - main stage
2 - pilot stage
3(c) - contraction cross section
L - orifice
e - effective value
p - spring chamber of main stage

4. User Defined Parameters

(1) cracking pressure in bar
(2) mass of pilot poppet in kg
(3) spring stiffness of pilot poppet in N/m
(4) viscosity coefficient of pilot poppet in NS/m
(5) diameter of pilot stage poppet seat in m
(6) area of pilot stage poppet seat in m m^2
(7) angle of pilot poppet face in degree
(8) volume of pilot stage chamber in m
(9) spring preload of main stage in N

(10) mass of main stage in kg
(11) spring stiffness of main stage in N/m
(12) viscosity coefficient of main stage NS/m
(13) diameter of inlet port of main stage in m
(18) area of inlet port of main stage in m^2
(14) angle of main stage spool face in degree
(15) diameter of main spring chamber in m
(16) area of main spring chamber in m^2
(17) clearance between spool and valve cover in m
(18) viscosity length of main stage in m
(19) orifice diameter in m
(20) initial pressure value of pilot stage in bar

5. Model Equations

The mathematical model developed for this relief valve includes one flow continuity equation of the pilot stage, one motion equation of the pilot poppet and one motion equation of the main poppet. They are given below respectively.

a) Flow continuity equation of pilot stage

$$\frac{V_2}{\beta}\frac{dP_2}{dt} = Q_L + A_p v_x - Q_2 - A_2 v_y \qquad \ldots\ldots (C.8)$$

From this equation, the pressure behaviour of the pilot chamber can be described by:

$$\frac{dP_2}{dt} = \frac{\beta}{V_2}\left(Q_L + A_p X - Q_2 - A_2 Y \right) \qquad \ldots\ldots (C.9)$$

We will now assume that the flow through the orifice is proportional to the square root of the pressure drop across it, i.e.

$$Q_L = K_{or} \sqrt{(P_1 - P_2)}$$

where $K_{or} = \frac{\pi}{4} C_d \ d_o^2$ - flow coefficient of orifice

Q_2 - flow through the orifice of pilot poppet

$$Q_2 = \pi C_d \ D_2 Y \ Sin(\alpha_2) \sqrt{2(P_2 - P_3)/\rho} \qquad \ldots\ldots (C.10)$$

b) Equation of motion for pilot poppet valve. Since the transient flow forces acting on the poppet valves are assumed to be ignored, the equation of motion for pilot poppet valve can be described by:

$$M_2 \frac{d^2Y}{dt^2} + f_2 \frac{dY}{dt} + K_2 Y = P_2 A_2 - P_{cr} \ A_2 - \frac{\rho Q_2^2}{C_d A_c} Cos(\alpha_2) Y \qquad \ldots\ldots (C.11)$$

where $\frac{\rho Q_2^2}{C_d A_c} Cos(\alpha_2) Y$ is steady state flow force acting on the pilot poppet valve. Assume that the areas of the contracted jet on opening and closing are equal in the pilot poppet orifice, i.e.

$$A_{open} = \pi C_d \ D_2 Y \ Sin(\alpha_2) = A_c$$

and combining equations (C.10) and (C.11), we obtain

$$M_2 \frac{d^2Y}{dt^2} + f_2 \frac{dY}{dt} + K_{e2} Y = P_2 A_2 - P_{cr} \ A_2 \qquad \ldots\ldots (C.12)$$

where K_{e2} - effective spring stiffness

$$K_{e2} = K_2 + \pi C_d \ D_2 Sin(2\alpha_2)(P_2 - P_3)$$

c) Equation of motion for main poppet valve. Similarly, assume the transient flow force acting on the main poppet valve is ignored and the areas of opening and contraction cross sections are equal in the main poppet orifice, the equation of motion for main poppet valve can be expressed in the same form:

$$M_1 \frac{d^2X}{dt^2} + f_1 \frac{dX}{dt} + K_{e1} X = P_1 A_1 - P_2 A_p - F_1 \qquad \ldots\ldots (C.13)$$

where K_{e1} - equivalent spring stiffness

$$K_{e1} = K_1 + \pi C_d D_1 Sin(2\alpha_1)(P_1 - P_3)$$

(d) Consideration of end stops

The motion of the poppets between the end stops, i.e. $0 < X < X_{max}$ and $0 < Y < Y_{max}$, is described by the equations of motion shown in equations (C.12) and (C.13). The mathematical end stop regions are defined for $Y >$ stroke, $X >$ stroke and $Y < 0$, $X < 0$, but in the practical conditions the poppet velocity must be zero when the poppet hits the end stop. Hence the mathematical model of end stops for two stage relief valve poppets needs to be established so that the complete motion characteristics of the main stage and the pilot stage can be described.

Actually the model of end stops can be described by non-linear functions and solved easily using logical conditional statements. For a pilot poppet the mathematical equation of end stop is expressed using the following non-linear functions:

$$\text{if } Y > Y_{max} \text{ or } Y < 0 \qquad \dot{v}_y = 0$$

$$\text{if } Y < 0 \quad \text{then} \quad Y = 0 \quad \text{and} \quad v_y = 0$$

$$\text{if } Y > Y_{max} \quad \text{then} \quad Y = Y_{max} \quad \text{and} \quad v_y = 0$$

similarly, for the main spool the mathematical model of the end stop is given by:

if $X > X_{max}$ or $X < 0$ $\dot{v}_x = 0$

if $X < 0$ then $X = 0$ and $v_x = 0$

if $X > X_{max}$ then $X = X_{max}$ and $v_x = 0$

6. Model Linking

The model has 2 external links which receive pressure and supply flow, and has 3 internal links. Two internal links are designed for displaying the displacements and appropriate velocities of the main poppet and pilot poppet, and the third link is designed for displaying the pressure value of pilot stage. The link diagram for model PCR1 is shown in figure C-5.

Model Inputs: Pressure in bar. EFFORT

Model Outputs: Flow rate in L/s FLOW

A typical sub-circuit containing model PCR1 together with its block diagram is shown in figure C-6.

The link ordering during program generation must correspond to the port ordering shown in figure C-5. The only acceptable input to the program generator for the sub-circuit shown in figure C-6 is:

PCR101 06 07 08 09 10

DERT - AN EFFORT VARIABLE DUTY CYCLE MODEL

1. Introduction

This model can be used to give a constant or variable input to any other model which requires an EFFORT value as input. The duty cycle comprises of from 1 to 12 stages, each stage remaining constant or varying linearly with time. The number of stages required is set by the user. The magnitude of the variable at the end of each stage is equal to its initial value for the next stage. The duty cycle retains the final linear function of the duty cycle if a simulation exceeds the time corresponding the end of the final stage[7].

The user may select one of the following variables to represent the duty cycle.

(i) Pressure
(ii) Torque
(iii) Displacement
(iv) Fractional displacement or swash fraction
(v) Angle
(vi) Voltage
(vii) Force
(viii) Temperature

The model can only be used for simulations using Runge-Kutta or Kutta-Merson integration subroutines.

2. Assumption

The output variable of this model is assumed to be a pseudo-state variable so that it can join the error test of determining integration time step, and the possible unstable problem of numerical solutions caused by the non-state variable in duty cycle model can be avoided.

3. Nomenclature

V - generalised effort variable

V_n - generalised effort variable at end of stage n

V_{n+1} - generalised effort variable at end of stage n+1

t - time in seconds

$t_1, t_2, \ldots\ldots t_n, t_{n+1}$ - times at end of stages 1,2, ... n, n+1 in seconds.

4. User defined parameters

(i) specific output variable.

(ii) number of stages in the duty cycle.

(iii) magnitude of output variable at the beginning of the first stage.

(iii) magnitude of output variable at the end of the first stage.

(iv) magnitude of output variable at the end of each stage.

(v) time at which each stage ends in seconds.

5. Model equations

According to assumption mentioned earlier, the output variable V is a pseudo-state variable whose derivative is equal to zero. i.e.

$$\frac{dV}{dt} = 0$$

The output variable at time t in stage n+1 of a duty cycle is actually given by

$$V = V_n + \frac{(V_{n+1} - V_n)}{(t_{n+1} + t_n)} (t - t_n)$$

Figure C-7 shows a typical duty cycle comprising 4 stages.

6. Model linking

The model has one external link providing a generalised effort variable. The link diagram for model DERT is shown in figure C-7.

Model inputs:

None.

Model outputs:

Effort variable, one of the following:

(i) Pressure
(ii) Torque
(iii) Displacement
(iv) Fractional displacement or swash fraction
(v) Angle
(vi) Voltage
(vii) Force
(viii) Temperature

PIR5 - FRICTIONLESS PIPE WITH AIR RELEASE AT LOW OPERATING PRESSURES.

1. Introduction

PIR5 models the dynamic behaviour of a pipe taking into account the compliance of the fluid and the pipe walls. Fluid friction and fluid inertia effects are not taken into account. The model includes a representation of air release when the pipe is operating at low pressures. At low operating pressures, the model calculates the proportion by volume of the air released from solution and the effect of this free air on the effective bulk modulus of the fluid. The model allows the specification of the pressure at which the fluid is saturated with air, that is the pressure below which air release will occur and will give a reasonably accurate representation of cavitation in most cases. The model receives fluid flows as input and supplies pressure to adjoining models[7].

The model can only be used for simulations using Runge-Kutta or Kutta-Merson integration subroutines.

2. Assumptions

(i) The air released from solution will be expanded polytropically with a constant index of 1.4.

(ii) The pipe fluid flow rates provided as model inputs will be valid during cavitation. These flows will be calculated in other models on the basis of single phase liquid flow. In fact two phase flow will exist during cavitation and the model will be in error in the event of flow from a cavitating pipe into another cavitating pipe through a control valve.

(iii) All flow from components connected to a pipe contains the same proportions of air and fluid that exist in the pipe.

(iv) Cavitation will cease immediately when flow sufficient to compress the free air to the saturation condition, has re-entered the pipe. This should be reasonably accurate for conditions where fluid is continually being replenished as in the case of a hydraulic transmission system.

(v) The release of fluid vapour will only become significant ubder extreme cavitation conditions and can be ignored.

(vi) Pressure drops due to pipe friction are assumed to be insignificant and are therefore not accounted for in this model.

(vii) Transient pressure variations due to fluid inertia (momentum effects) are assumed to be insignificant and are therefore not accounted for in this model.

3. Nomenclature

a' - Proportion (measured by volume) of free air at working pressure
a - Proportion (measured by volume at STP) of free air at working pressure
β_a - Bulk modulus of air in bar
β_e - Effective bulk modulus of fluid or fluid-air combination in bar
β_f - Bulk modulus of fluid in bar
d_0 - Proportion (measured by volume) of air dissolved in fluid at STP
$f(P)$ - Function relating to the fluid compressibility
d_{sat} - Proportion (measured by volume) of air dissolved in
P - Pipe pressure in bar
P_{sat} - Pressure at which fluid is saturated with air in bar
Q - Sum of flow inlet to pipe - flow outlet from pipe in L/s
Q'_a - Rate of air release measured at STP in L/s
Q_a - Rate of air release measured at working pressure in L/s
Q_{in} - Net flow rate into pipe in L/s
n - Polytropic index
V_a - Volume of air in pipe control volume in L
V - Volume of pipe in L

4. User defined parameters

(i) pipe internal diameter in mm.

(ii) pipe length in m.

(iii) Either effective bulk modulus in bar or pipe wall thickness and internal diameter in mm and pipe material (in order to determine the compliance of the pipe). The pipe materials at present available are steel, tungum, cupro-nickel and flexible hose.

(iv) pressure at which oil is saturated with air in bar.

(v) proportion of air by volume, dissolved in oil at standard temperature and pressure.

(vi) initial pipe pressure in bar.

5. Model equations

At all operating pressures, the behaviour of the pipe is described by the general continuity equation.

$$\frac{dP}{dt} = \frac{\beta}{V} \Sigma Q \qquad \ldots\ldots (C.14)$$

Where β is an effective bulk modulus and is computed by

$$\beta = \beta_f \qquad \text{if} \quad P > P_{sat}$$

$$\beta = \frac{1}{\frac{1}{\beta_f} + \frac{d_0 (P_{sat} - P)}{n (P+1)^2}} \qquad \text{if} \quad P \leqslant P_{sat}$$

The minimum working pressure is set to be -0.999 bar, i.e.

$$\text{if} \quad P < -1 \text{ bar} \qquad P = -0.999 \text{ bar}$$

6. Model linking

The model has 8 external links which receive flow and supply pressure. The external links may be specified in any order. The link diagram for model PIR5 is shown in figure C-9.

Model inputs:

Flow (L/s) (available on up to 8 ports). FLOW

Model outputs:

Pressure in bar (available on up to 7 ports). EFFORT

A typical sub-circuit containing model PIR5, together with its block diagram is shown in figure C-10.

The link ordering during program generation must correspond to the port ordering shown in figure C-9. The only acceptable input to the program generator for the sub-circuit shown in figure C-10 is:

PIR501 08 07 09

LRR0- A ROTARY LOAD MODEL

1. Introduction

Model LRR0 is a model of a rotary load and it is used to simulate the instantaneous behaviour of a rotary load in which inertia, viscous friction, Coulomb friction, stiction and windage effects are represented. The model accepts an input angular speed from the shaft model and supplies an output load torque to the adjoining shaft model. A variable or constant external load torque may be provided by a duty cycle model DEDT.

2. Assumption

The stiction torque is assumed to be independent of the time spent at rest.

3. Nomenclature

f_v - Viscous friction coefficient in Nm/(rev/min)
f_w - Windage coefficient in Nm/(rev/min)
J - Moment of inertia of load in $Kg.m^2$
T - Applied torque in Nm
T_f - Coulomb friction torque in Nm
T_s - Stiction torque in Nm
T_c - Coulomb friction torque in NM
T_{EXT} - External variable or constant load torque in Nm
ω - Angular speed in rev/min

4. User defined parameters

(i) Moment of inertia of load in Kgm^2
(ii) Stiction torque in Nm
(iii) Coulomb friction torque in Nm
(iv) Viscous friction coefficient in Nm/(rev/min)
(v) Windage coefficient in $Nm/(rev/min)^2$

5. Model equations

The driving torque is given by the following equation:

$$T = T_{EXT} + SIGN((T_f + f_v|\omega| + f_w\omega^2),\omega) \qquad \ldots\ldots (C.15)$$

where

$$T_f = \begin{cases} T_c & |\omega| > 0 \\ T_s & |\omega| = 0 \end{cases}$$

6. Model linking

The model has a single external link which receives angular speed ω and supplies torque (T).

Model inputs:

Angular speed	EFFORT

Model outputs:

Torque	FLOW

The link diagram for model LRR0 is shown in figure C-11. A typical sub-circuit

containing LRR0 together with its block diagram is shown in figure C-12. The input to the program generator for this model is:

LRR001 02

MOA1 - A FIXED DISPLACEMENT HYDRAULIC MOTOR MODEL

1. Introduction

MOA1 is a model of a fixed displacement hydraulic motor. The model considers the dynamic behaviour of the motor and takes into account slip and case leakages and compressibility losses. The associated CETOP symbol is shown in figure C-13.

The model receives quantities inlet pressure, outlet pressure and torque from adjoining models and supplies inlet flow, outlet flow and angular velocity to adjoining models.

2. Assumptions

(i) The model operates at a constant displacement.
(ii) The coefficients for slip, compressibility, and frictional losses are assumed to be constant throughout the motor operating range.
(iii) The model assumes the case drain pressure, if appropriate, to be zero bar gauge.

3. Nomenclature

β - bulk modulus of fluid in bar
C_s - slip loss coefficient
C_{s1} - internal (cross-port) slip loss coefficient
C_{s2} - external (drain line) slip loss coefficient
C_f - motor friction loss coefficient
D_m - maximum motor displacement in L/rev
J_m - motor inertia in kgm^2

P_{in} - motor inlet pressure in bar

P_{out} - motor outlet pressure in bar

P_{tank} - tank pressure in bar

Q_{in} - motor inlet flow in L/s

Q_{out} - motor outlet flow in L/s

Q_c - compressibility flow in L/s

Q_s - leakage flow in L/s

Q_{s1} - internal (cross-port) leakage flow in L/s

Q_{s2} - external (drain) leakage flow in L/s

T_s - torque supplied by shaft in Nm

V_c - clearance volume in L

ω_m - angular velocity of motor in rev/min

μ - viscosity of fluid in Ns/m

4. User defined parameters

(i) Maximum motor displacement, Dm, in L/rev

(ii) Slip loss coefficient, Cs, due to differential pressure
(typical values are suggested)

(iii) Relative pressure dependent torque loss coefficient
(typical values are suggested)

(v) Relative viscous friction coefficient
(typical values are suggested)

(vi) Clearance volume, Vc, as a fraction of full displacement
(typical values are suggested)

(vii) Type of motor - (P)iston or (G)ear

5. Model equations

Torque Balance Equation

$$\frac{d\,\omega_m}{dt} = \frac{1}{J_m}\left(D_m \left(1 - C_f \right) - T_s - f_w\, \omega_m \right) \qquad \ldots\ldots \text{(C.16)}$$

Theoretical flow Qth

$$Q_{th} = D_m \omega_m \qquad \ldots\ldots (C.17)$$

Slip flow losses

The slip flow loss coefficient, Cs, is divided into two parts, Cs1, is the internal slip loss due to cross-port leakage and Cs2, the external slip loss to case and tank.

$$C_s = C_{s1} + C_{s2} \qquad \ldots\ldots (C.18)$$

Data from an extensive range of tests carried out on piston and gear units has been analysed giving the following broad relationships.

for piston units, $C_{s2} = 5\, C_{s1}$

for gear units, $\dfrac{C_{s2}}{5} = C_{s1}$

(a) Cross-port leakage loss Qs1

$$Q_{s1} = \frac{C_{s1}}{\mu} D_m \, (P_{in} - P_{out}) \qquad \ldots\ldots (C.19)$$

(b) External leakage to case and tank Qs2

$$Q_{s2} = \frac{C_{s2}}{\mu} D_m \, (P_{out} - P_{tand}) \qquad \ldots\ldots (C.20)$$

Compressibility loss Qc

$$Q_c = \left(\frac{V_c}{D_m} + 1 \right) \frac{P_{out} - P_{in}}{\beta} D_m \omega_m \qquad \ldots\ldots (C.21)$$

Net inlet flow Qin

$$Q_{in} = Q_{th} - Q_{s1} \qquad \ldots\ldots (C.22)$$

Net delivery flow Qout

$$Q_{out} = Q_{th} - Q_{s1} - Q_{s2} - Q_c \qquad \ldots\ldots (C.23)$$

6. Model linking

The model has 3 external links. The external link connected to port 1 receives inlet pressure and supplies inlet flow. The external link connected to port 2 receives outlet pressure and supplies outlet flow. The external link connected to port 3 receives torque and supplies angular speed. The link diagram for model MOA1 is shown in figure C-14.

Model inputs:

P_{in}	inlet pressure (bar)	EFFORT
P_{out}	outlet pressure (bar)	EFFORT
T_s	torque (Nm)	EFFORT

Model outputs:

Q_{in}	motor inlet flow (L/s)	FLOW
Q_{out}	motor outlet flow (L/s)	FLOW

ω_m angular velocity (rev/min) FLOW

A typical sub-circuit containing model MOR0, together with its block diagram is shown in figure C-15.

The link ordering during program generation must correspond to the port ordering shown in figure C-14. Thus the only acceptable input to the program generator for the sub-circuit shown in figure C-15 is:

MOA101 06 10 07

PMR0 - INSTANTANEOUS MODEL OF A DIESEL ENGINE

1. Introduction

PMR0 models the instantaneous behaviour of a diesel engine. This model is concerned with operation at speeds in excess of that at which maximum engine torque is acheived. It assumes a linear decrease in torque from maximum engine torque to the torque at maximum operating speed. The governed speed is specified by the user, and the governor droop is also required. By droop is meant the fall off in speed which occurs as the torque increases form zero to full rated torque.

The model receives angular speeds as input and supplies torque to adjoining models. The model can only be used for simulations using Runge-Kutta and Kutta-Merson integration subroutines.

2. Assumption

The characteristics representing maximum torque, speed droop and motoring are assumed to be linear.

3. Nomenclature

T_E - Diesel engine torque in NM
T_{MO} - Maximum negative output torque of engine in NM
T_{MX} - Maximum output torque of engine in NM
ω_0 - Angular speed associate with T_{MX} in rev/min
ω_D - Rated angular speed of governor in rev/min
ω - Rated angular speed of governor in rev/min
ω_E - Angular speed of shaft in rev/min

4. User defined parameters

(i) - maximum output torque of engine in NM
(ii) - angular speed associate with maximum output torque in rev/min
(iii) - rated angular speed of governor in rev/min
(iv) - maximum negative output torque of engine in NM

5. Model equations

(i) if $\omega_E \leqslant \omega_D$

$$T_E = T_{MX} - (\omega_E - \omega_0) \frac{0.16667\, T_{MX}}{(\omega_D - \omega_0)} \qquad \ldots\ldots (C.24)$$

(ii) if $\omega_E > \omega_D$

$$T_E = 0.8333\, T_{MX} - (\omega_E - \omega_0) \frac{0.8333\, T_{MX}}{\frac{\omega_D}{9}} \qquad \ldots\ldots (C.25)$$

where if $T_E < T_{MO}$ $T_E = T_{MO}$

Model Linking

The inputs and outputs of model PMR0 are listed below.

Model inputs:

Angular speed in rev/min FLOW

Model outputs:

Engine torque in NM EFFORT

The model has one external link which receive angular speed and supply torque. The link diagram for model PMR0 is shown in figure C-16. A typical sub-circuit containing model PMR0, together with its block diagram is shown in figure C-17.

The link ordering during program generation must correspond to the port ordering shown in figure C-16. The only acceptable input to the program generator for the sub-circuit shown in figure C-17 is:

PMR001 01

ORR0 - A LONG ORIFICE RESTRICTOR MODEL

1. Introduction

ORR0 models the behaviour of a long orifice restrictor, used for the control of fluid flow. The design symbol for a long orifice restrictor is shown in figure C-18. The model receives inlet and outlet pressure and supplies inlet and outlet flow to adjoining models. Account is taken of fluid compressibility in calculating inlet flow. Power loss in the orifice can also be computed.

2. Assumptions

(i) The flow-pressure characteristics of the valve is linear.
(ii) Fluid compressibility is accounted for in the model.

3. Nomenclature

P_{in} - inlet pressure in bar
P_{out} - outlet pressure in bar
Q_{in} - inlet flow in L/sec
Q_{out} - outlet flow in L/sec
W - power loss in watts
d - orifice diameter in mm
l - orifice length in mm
C_L - amending coefficient of the valve
β - bulk modulus in bar
ρ - fluid density in Kg/m^3

4. User defined parameters

(i) an orifice diameter in mm
(ii) an orifice length in mm

5. Model equations

For a long orifice the linear law is assumed. The flow is calculated using the equation:

$$Q_{out} = \frac{\pi \, d_o^4}{128 \, \mu \, l_o \, C_L} \qquad \ldots\ldots (C.26)$$

The inlet flow to the orifice is calculated taking into account compressibility between the high and low pressure sides of the orifice. By convention the inlet flow to the valve is taken to be opposite in sign to the outlet flow:

$$Q_{in} = -Q_{out} \left(1 - \frac{(P_{in} - P_{out})}{\beta} \right) \qquad \ldots\ldots (C.27)$$

The power loss across the orifice is given by

$$W = P_{in} \, Q_{in} - P_{out} \, Q_{out} \qquad \ldots\ldots (C.28)$$

6. Model linking

The model has 2 external links. The external link connected to port 1 receives inlet pressure and supplies inlet flow. The external link connected to port 2 receives outlet pressure and supplies outlet flow. The link diagram for model ORR0 is shown

in figure C-19.

Model inputs:

Inlet pressure in bar.

Outlet pressure in bar.

Model outputs:

Inlet flow in L/sec.

Outlet flow in L/sec.

A typical sub-circuit containing model ORR0, together with its block diagram is shown in figure C-20.

The link ordering during program generation must correspond to the port ordering shown in figure C-19. The only acceptable input to the program generator for the sub-circuit shown in figure C-20 is:

ORR001 04 05

SH00 - DYNAMIC MODEL OF A CIRCULAR SHAFT WITHOUT TORQUE LOSS

1. Introduction

SH00 models the dynamic behaviour of a shaft taking into account the rigidity of the metal materials and the polar moment of inertia. The torque loss and the inertia (or mass) effects of the shaft are not taken into account. The model receives the angular speeds as input and supplies torque to adjoining models.

2. Model Assumption

The torque loss dependent mechanical friction is ignored.

3. Nomenclature

T_s - Shaft torque in NM
ω_1 - Shaft input angular speed in 1/S
ω_2 - Shaft output angular speed in 1/S
G - Modulus of rigidity for shaft material
I - Polar moment of inertia in M^4
L - Shaft length in M

4. User defined parameters

(i) - initial shaft torque in NM
(ii) - modulus of rigidity for shaft material
(iii) - polar moment of inertia in M^4
(iv) - shaft length in M

5. Model equations

The angular speed continuity differential equation for a shaft with two supplies is given by:

$$\frac{dT_s}{dt} = \frac{G\ I}{L}\left(\omega_1 - \omega_2\right) \qquad \ldots\ldots (C.29)$$

6. Model Linking

The inputs and outputs of model SH00 are listed below.

Model inputs:

Angular speed (1/S) (available on up to 2 ports). FLOW

Model outputs:

Torque in NM (available on up to 2 ports) EFFORT

The model has two external links which receive angular speeds and supply torque. The external links may be specified in any order. The link diagram for model SH00 is shown in figure C-21.

A typical sub-circuit containing model SH00, together with its block diagram is shown in figure C-22.

The link ordering during program generation must correspond to the port ordering shown in figure C-21. The only acceptable input to the program generator for the sub-circuit shown in figure C-22 is:

SH0001 10 11

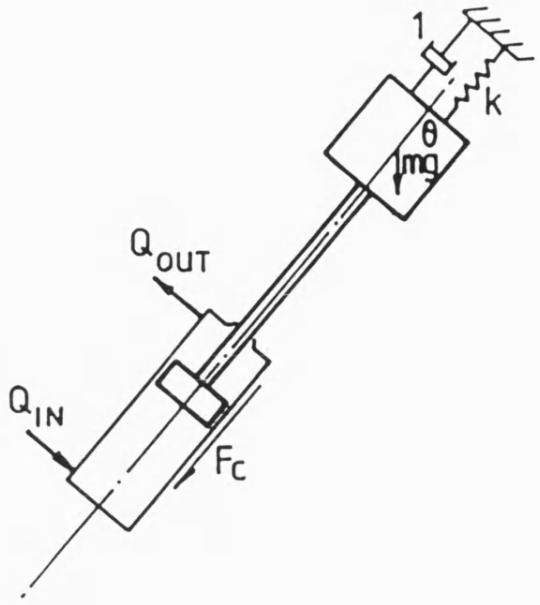


FIG.C-1 Linear actuator with load

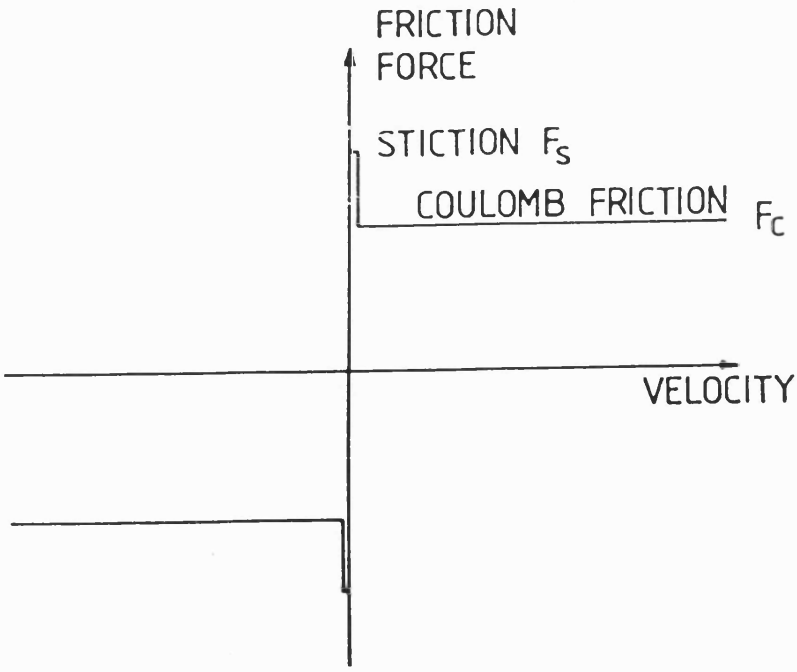


FIG.C-2 Stick/Slip friction characteristic

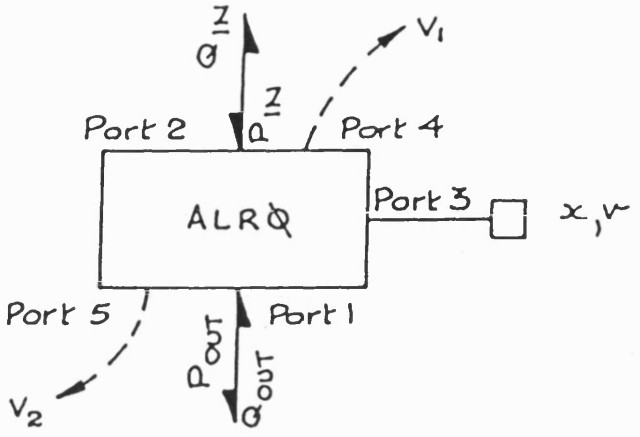


FIG.C-3 Link diagram for ALRØ, a linear actuator with load

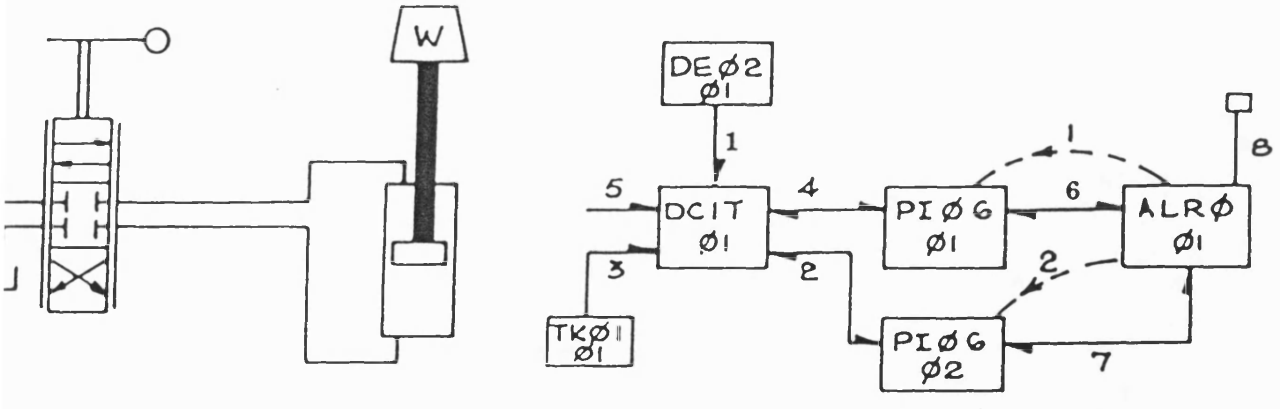


FIG.C-4 Test circuit and corresponding block diagram.

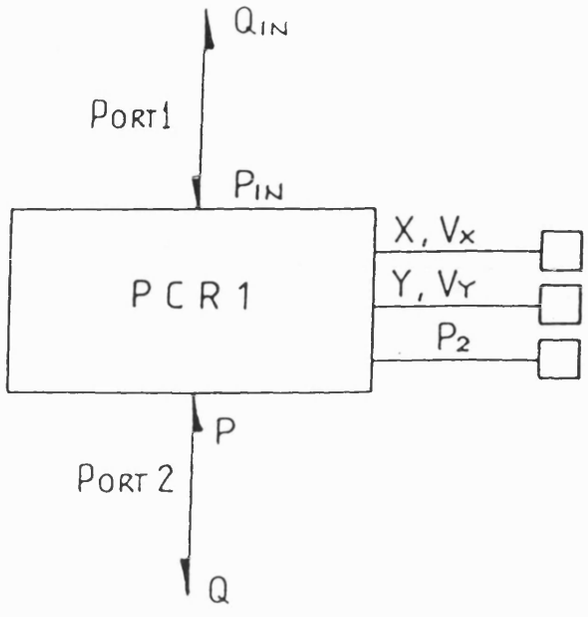


FIGURE C-5 LINK DIAGRAM FOR MODEL PCR1
A TWO STAGE RELIEF VALVE

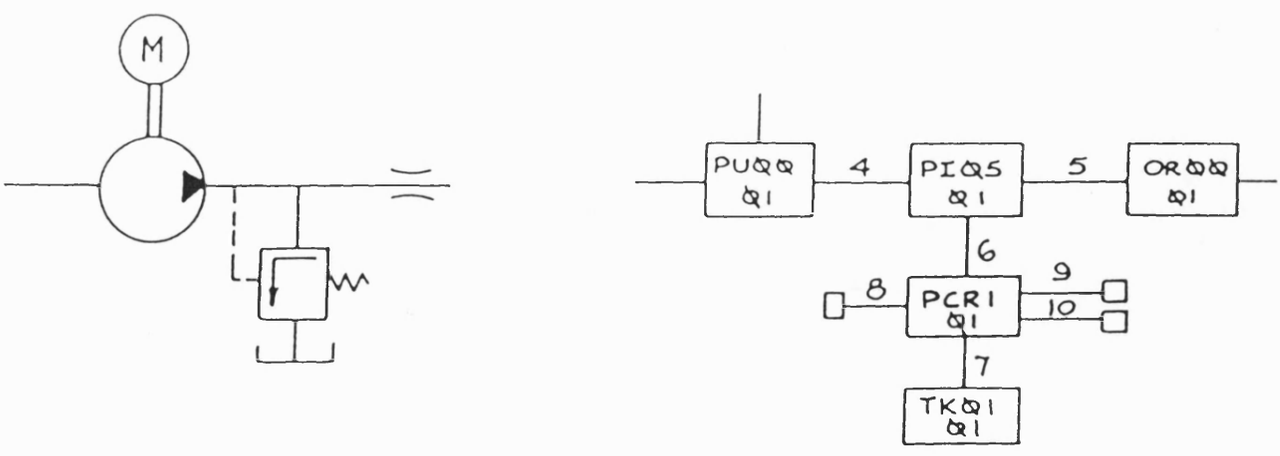


FIGURE C-6 SUB-CIRCUIT BLOCK DIAGRAM FOR SIMULATION
CORRESPONDING TO MODEL PCR1

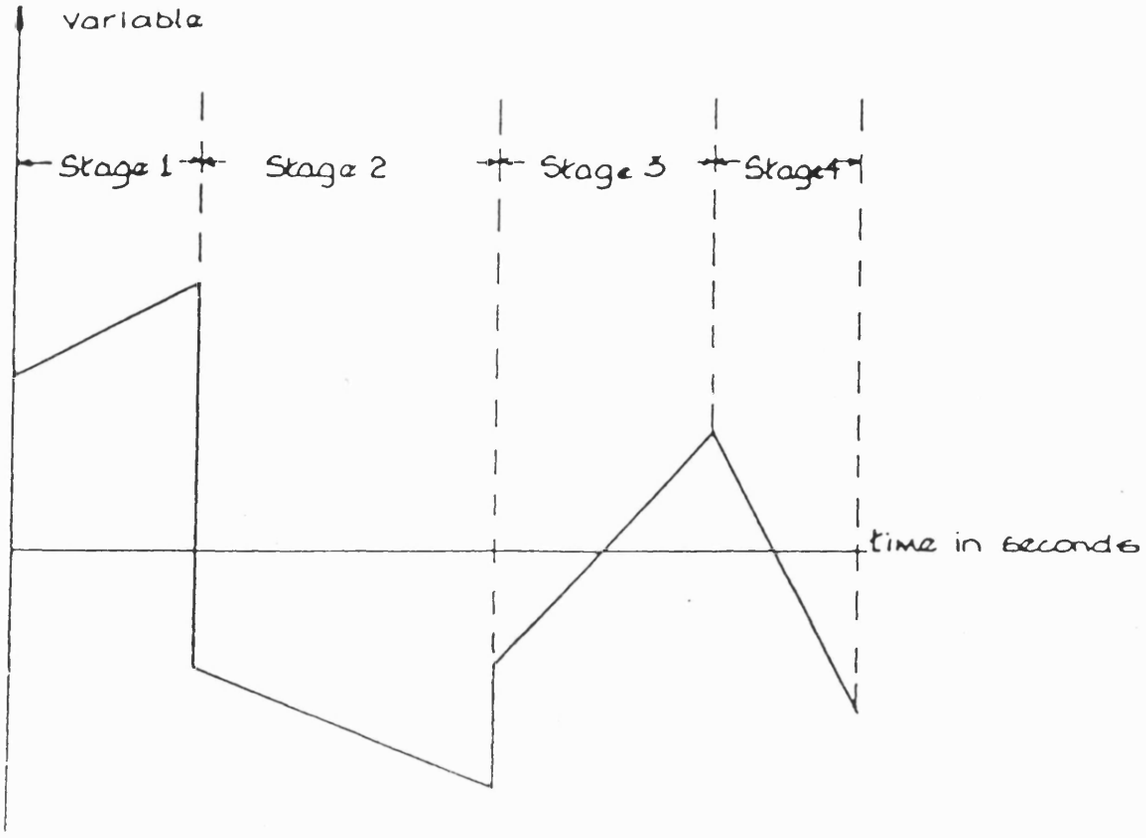
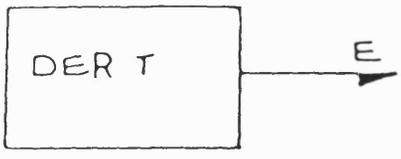
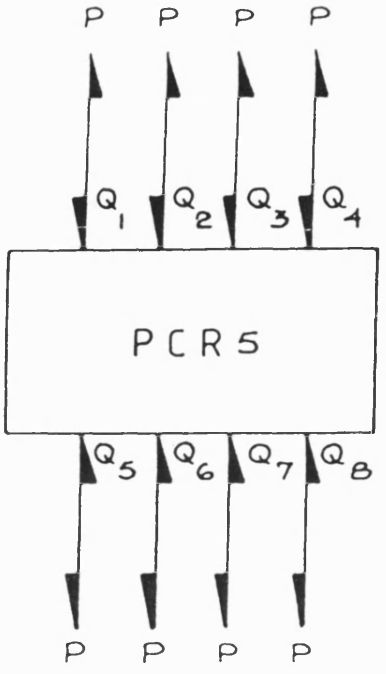


Figure C-7 Typical time dependent duty cycle



EFFORT : E - PRESSURE
TORQUE
DISPLACEMENT
FRACTIONAL DISPLACEMENT
ANGLE
VOLTAGE
FORCE
TEMPERATURE

Figure C-8 Link diagram for duty cycle model DERT



EFFORTS - P

FLOWS - Q_1 TO Q_8

FIGURE C-9 LINK DIAGRAM FOR PIR5 - MODEL OF A PIPE

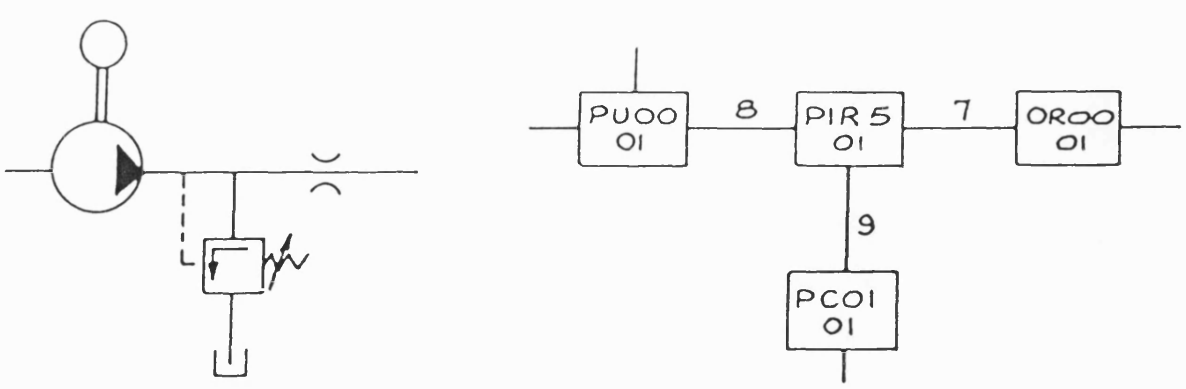


FIGURE C-10 SUB-CIRCUIT AND BLOCK DIAGRAM FOR SIMULATION USING MODEL PIR5

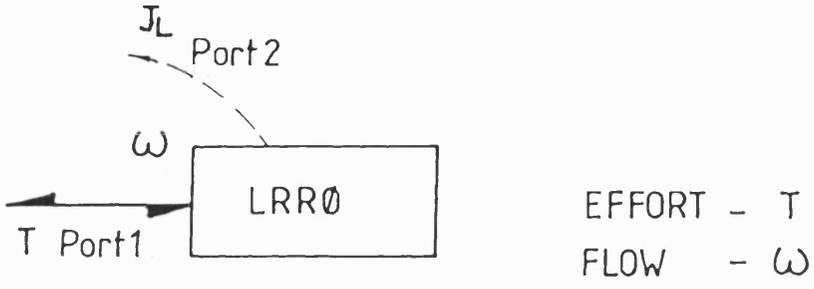


FIG. C-11 LINK DIAGRAM FOR LRRØ - A ROTARY LOAD

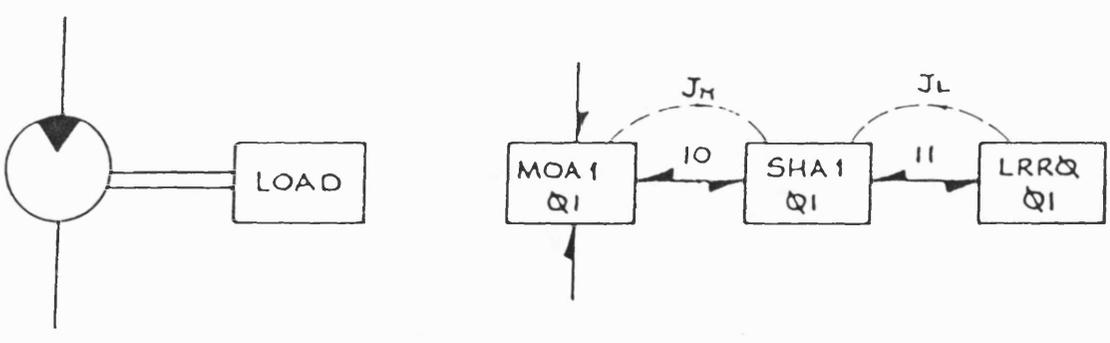


FIG. C-12 HYDRAULIC SUB-CIRCUIT AND CORRESPONDING BLOCK DIAGRAM

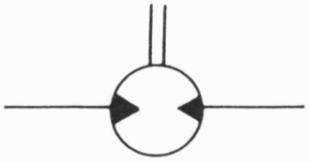
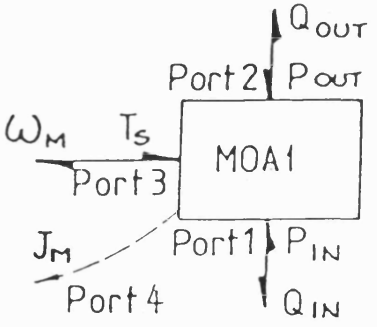


FIG C-13 CETOP symbol for hydraulic motor



EFFORTS P_{IN}, P_{OUT}, ω_M T_S

FLOWS Q_{IN}, Q_{OUT}, ω_M

FIG C-14 Link diagram of MOA1

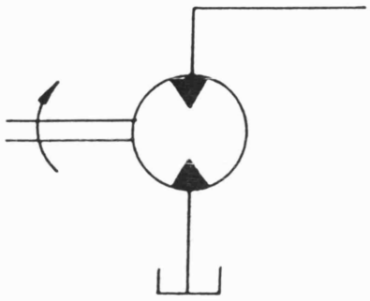


FIG C-15(a) Example of a circuit using MOA1

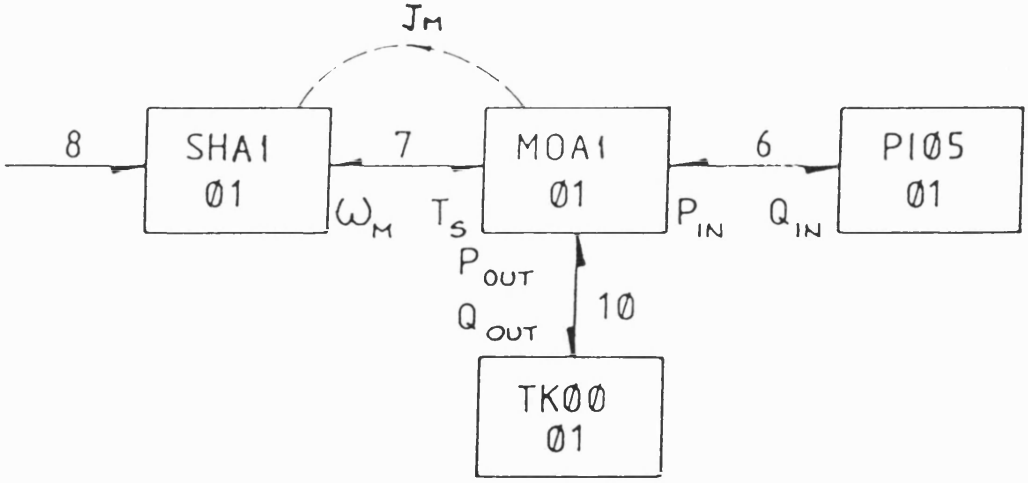
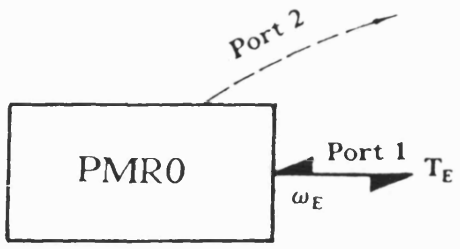


FIG. C-15(b) Block diagram for simulation of sub-circuit using MOA1



EFFORT: T_E
FLOW: ω_E

Figure C-16 Link diagram for PMR0 - an instantaneous model of a diesel engine

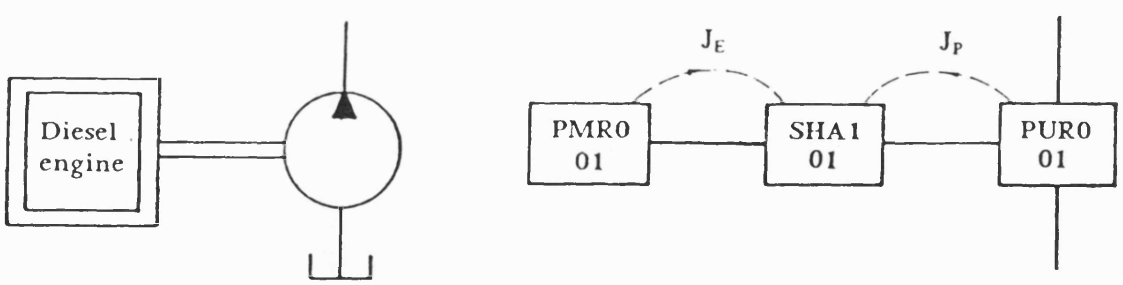


Figure C-17 Subcircuit and block diagram for simulation using model PMR0

FIG. C-18 DESIGN SYMBOL FOR AN ORIFICE RESTRICTOR

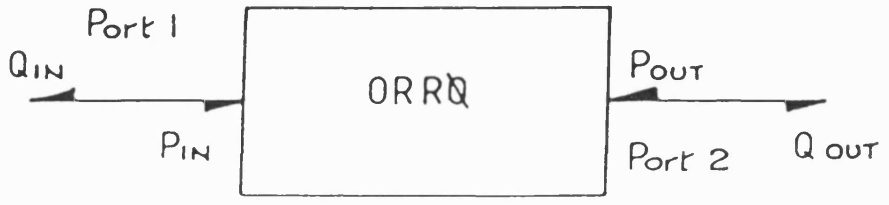


FIG. C-19 LINK DIAGRAM FOR MODEL ORRQ — A LONG ORIFICE

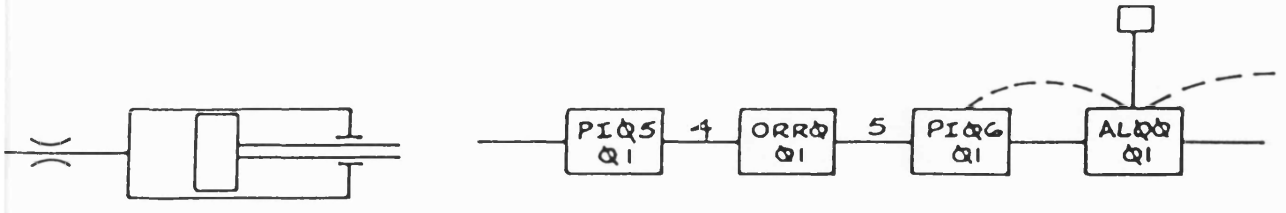
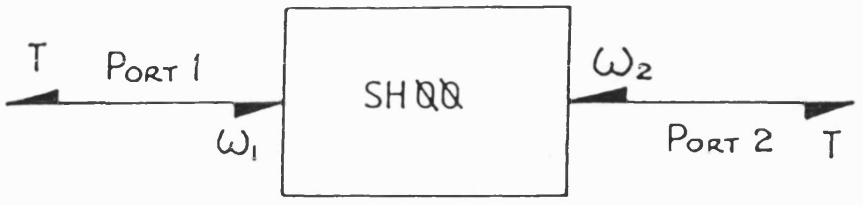


FIG. C-20 SUB-CIRCUIT AND CORRESPONDING BLOCK DIAGRAM



EFFORT : T
FLOWS : ω_1 , ω_2

FIGURE C-21 LINK DIAGRAM FOR SHAFT MODEL SHQQ

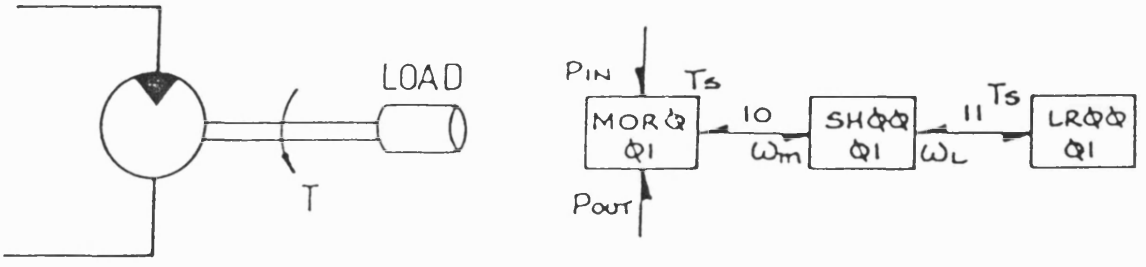


FIGURE C-22 SUB-CIRCUIT AND BLOCK DIAGRAM FOR SIMULATION USING MODEL SHQQ

APPENDIX D

LISTING OF SINGLE STEP INTEGRATION SUBROUTINES

CONTENTS

N.B. These integration subroutines are available in the HASP model library.

LISTING OF RK4 INTEGRATION SUBROUTINE

```
      SUBROUTINE RK4(N,T,TEND,TAB,Y,DY,YC,Y1,AUX,OUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TSTEP/HT
      DIMENSION Y(N),DY(N),YC(N),Y1(N),AA(4)
C ***************************************************************************************************
C INTEGRATION SUBROUTINE FOR HASP. USES RUNGE-KUTTA'S METHOD OF ORDER 4.
C METHOD INTEGRATES A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS OVER A RANG
C                  HAS PROVISION FOR FIXED STEP
C ***************************************************************************************************
C
C                          SCHOOL OF ENGINEERING
C                          UNIVERSITY OF BATH
C                          CLAVERTON DOWN
C                          BATH
C
C     DEVELOPED AND COMMENTS ADDED:    L.M. WANG
C     DATE:                    23-OCT-86
C
C ***************************************************************************************************
C
C VARIABLES IN ARGUMENT LIST:
C
C      N     -  NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS TO BE SOLVED
C      HT    -  SIMULATION TIME STEP IN SECONDS
C      T     -  SIMULATION TIME IN SECONDS
C      TEND  -  OVERALL SIMULATION TIME IN SECONDS
C      TAB   -  PRINT INTERVAL IN SECONDS
C      Y     -  ARRAY (OF SIZE N) OF STATE VARIABLES. STORES THE RESULTS
C               IN DATA FILE CADRES.DAT(FOR PLOTTING). USER HAS AN OPTION
C               TO PRINT RESULTS AS SIMULATION PROGRESSES.
C      DY    -  USED TO STORE  DERIVATIVES OF FUNCTIONS
C      YC,Y1 -  INTERMEDIATE CALCULATION VALUES OF STATE VARIABLES
```

```
C     AUX   -  SUBROUTINE WRITTEN BY THE HASP PROGRAM GENERATOR TO PROVIDE
C              A CALLING LIST FOR THE COMPONENT MODEL SUBROUTINES
C     OUT   -  SUBROUTINE FOR OUTPUTTING RESULTS OF SIMULATION
C
C ***** VARIABLES USED IN THE INTEGRATION SUBROUTINE
C
C     LIMIT -  LOGICAL CONTROL INDICATOR IN SUBROUTINE
C              LIMIT = 0 : INITIALISATION PROCESS
C              LIMIT = 1 : INTERMEDIATE CALCULATION PROCESS
C              LIMIT = 2 : WHEN AN INTEGRATION STEP HAS COMPLETED
C                          SUCCESSFULLY.
C     AA(I) -  COEFFICIENTS OF ITERATIVE EXPRESSION FOR INTEGRATION
C
C ***********************************************************************************************
C ***** THE OUTPUT SUBROUTINE IS CALLED TO PRINT OUT INITIAL CONDITIONS
C
      LIMIT=0
      TABC=TAB
      CALL OUT(T,Y,N,TAB,LIMIT)
C
C **** BEGIN TRIAL STEP
C
  140 T= T + HT
C
C ***** LIMIT SET TO 1 FOR INTERMEDIATE INTEGRATION PROCESS
C
      LIMIT=1
      AA(1)=HT/2.
      AA(2)=AA(1)
      AA(3)=HT
      AA(4)=HT
      DO 1 I=1,N
    1 Y1(I)=Y(I)
      DO 3 K=1,3
        DO 2 I=1,N
          YC(I)=Y1(I)+AA(K)*DY(I)
    2   Y(I)=Y(I)+AA(K+1)*DY(I)/3.
C
C ***** AUX IS CALLED TO OBTAIN THE DERIVATIVE VALUES.
```

```
C ***** STORED IN ARRAY DY(I)
C
  3    CALL AUX(DY,YC,T,N,LIMIT)
       DO 4 I=1,N
  4    Y(I)=Y(I)+AA(1)*DY(I)/3.
C
C ***** THE STEP IS COMPLETED. LIMIT IS SET 2 (IN MODELS AS WELL).
C ***** THE CALL TO AUX (CALLS MODELS) IS MADE TO
C *****    (1) OBTAIN THE DERIVATIVE VALUES.
C *****    (2) SOLVE DISCONTINUITIES BY "IF STATEMENTS".
C
      LIMIT=2
      CALL AUX(DY,Y,T,N,LIMIT)
C
C ***** IF CALCULATION TIME IS LESS THAN PRINT TIME, GOTO 140
C ***** ELSE PRINT TIME PLUS AN INTERVAL TAB.
C
      IF(T.LT.TABC) GOTO 140
      TABC = TAB + T
C
C ***** THE OUTPUT SUBROUTINE (OUT) IS CALLED TO STORE THE SIMULATION RESULTS
C ***** IN THE RESULTS FILE (CADRES.DAT) AND IF REQUESTED TO PRINT OUT THE
C ***** RESULTS ON THE SCREEN. NORMALLY, THE RESULTS ARE PRINTED OUT AT THE
C ***** USER-DEFINED PRINT INTERVALS.
C
      CALL OUT(T,Y,N,TAB,LIMIT)
C
C ***** IF CALCULATION TIME EQUALS OVERALL SIMULATION TIME, RETURN TO
C ***** MAIN SEGMENT
C
      IF(T.LT.TEND) GOTO 140
      END
```

LISTING OF KUTMER INTEGRATION SUBROUTINE

```
C ***************************************************************************************************
C INTEGRATION SUBROUTINE FOR HASP. USES RUNGE-KUTTA-KUTMER'S METHOD OF ORDE
C METHOD INTEGRATES A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS OVER A RANC
C                HAS PROVISION FOR INTERVAL HALVING
C ***************************************************************************************************
C
C                        SCHOOL OF ENGINEERING
C                        UNIVERSITY OF BATH
C                        CLAVERTON DOWN
C                        BATH
C
C    DEVELOPED AND COMMENTS ADDED:   L.M. WANG
C    DATE:                   25-APRIL-87
C
C ***************************************************************************************************
C
C VARIABLES IN ARGUMENT LIST:
C
C     N    - NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS TO BE SOLVED
C     HT   - SIMULATION TIME STEP IN SECONDS
C     T    - SIMULATION TIME IN SECONDS
C     TEND - OVERALL SIMULATION TIME IN SECONDS
C     TAB  - PRINT INTERVAL IN SECONDS
C     Y    - ARRAY (OF SIZE N) OF STATE VARIABLES. STORES THE RESULTS
C            IN DATA FILE CADRES.DAT(FOR PLOTTING). USER HAS AN OPTION
C            TO PRINT RESULTS AS SIMULATION PROGRESSES.
C     DY   - USED TO STORE DERIVATIVES OF FUNCTIONS
C     TOL  - INTEGRATION TOLERANCE USED FOR ERROR TEST (SET IN SUBROUTINE
C            CONTRL)
C     AUX  - SUBROUTINE WRITTEN BY THE HASP PROGRAM GENERATOR TO PROVIDE
C            A CALLING LIST FOR THE COMPONENT MODEL SUBROUTINES
C     OUT  - SUBROUTINE FOR OUTPUTTING RESULTS OF SIMULATION
```

```
C
C ***** VARIABLES USED IN THE INTEGRATION SUBROUTINE
C
C      LIMIT  - LOGICAL CONTROL INDICATOR IN SUBROUTINE
C               LIMIT = 0 : INITIALISATION PROCESS
C               LIMIT = 1 : INTERMEDIATE CALCULATION PROCESS
C               LIMIT = 2 : WHEN AN INTEGRATION STEP HAS COMPLETED
C                           SUCCESSFULLY.(OR INITIALISATION PROCESS)
C
C      DELTAX -  X-RANGE FOR WHICH THE INTEGRATION IS TO BE DONE IN ONE CALL
C      DLTAY  -  DIFFERENCE BETWEEN CURRENT TIME STATE VARIABLES AND LAST
C                TIME STATE VARIABLES ( Y(I)-YN(I) )
C      DY0,DY1,DY2
C             - INTERMEDIATE CALCULATION VALUES OF DERIVATIVES
C      HEST   -  AN ESTIMATED VALUE OF HT WHICH IS PROVIDED TO START OFF
C                THE SUBROUTINE.
C      HOUT   -  EXCHANGE VARIABLE OF ESTIMATED STEP LENGTH HEST
C      K      -  NUMBER OF INTEGRATION  STEPS TAKEN IN THE RANGE DELTAX
C      T0     -  INITIAL INTEGRATION TIME OF EACH INTERVAL
C      TABC   -  PRINT TIME IN SECONDS
C      Y0,Y1,Y2
C             - INITIAL (OR PREVIOUS) VALUES AND ESTIMATES OF STATE
C               VARIABLES
C      YABC   -  STATE VARIABLES AT PRINT TIME POINT
C      YN     -  LAST COMPLETED STATE VARIABLES
C
C ***********************************************************************************************
C
      SUBROUTINE KUTMER(N,T,TEND,TAB,Y,DY,TOL,AUX,OUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TSTEP/HT
      DIMENSION Y(N),DY(N),Y0(50),Y1(50),Y2(50),DY0(50),DY1(50),
     +          DY2(50),YN(50),YABC(50),DLTAY(50),ERROR(50),DYTEMP(50)
      LOGICAL B1,B2
C
C **** ASSIGNMENT FOR AN ESTIMATED VALUE OF HT
C
      HEST=HT
C
```

```
C **** ASSIGNMENT FOR MAXIMUM AND MINIMUM STEP LENGTH
C
      HTMAX=1.D30
      HTMIN=1.D-6
C
C **** THE OUTPUT SUBROUTINE IS CALLED TO PRINT OUT INITIAL CONDITIONS
C
      LIMIT=0
      TABC=TAB
      CALL OUT(T,Y,N,TAB,LIMIT)
C
C **** BEGIN TRIAL STEP
C
  140  CONTINUE
C
C **** REASIGN THE VALUES OF T0, K
C
      T0=T
      K=0
C
C **** LET INTEGRATION STEP LENGTH HT BE THE ESTIMATED VALUE OF HT
C
      HT=HEST
C
C **** IF FIRST ENTRY THEN INITIALISE Y0 TO Y
C
      DO 200 I=1,N
        Y0(I)=Y(I)
  200  CONTINUE
C
C **** START INTEGATION LOOP IN ONE TIME-RANGE (DALTAX)
C
  102  B1=.FALSE.
      B2=.FALSE.
      DELTAX=1.5D0*HT
      HOUT=HT
C
C **** ERROR TEST OF DETERMINING LOGICAL VARIABLES
C
```

```
      TEST=(T+DELTAX-T0-HT)/(DELTAX+1.D-25)
      IF(TEST.GT.1.D-6)GOTO 101
      IF(TEST.LT.-1.D-6) THEN
       B1=.TRUE.
       HT=DELTAX-T0+T
       GOTO 101
      END IF
      B2=.TRUE.
C
C **** COMPUTE FIVE APPROXIMATIONS TO Y
C **** LET  LIMIT=1
C
  101  LIMIT=1
      IF(HT.GT.HTMAX) HT=HTMAX
      CALL AUX(DY0,Y0,T0,N,LIMIT)
      DO 201 I=1,N
        Y1(I)=Y0(I)+HT/3.0*DY0(I)
  201  CONTINUE
      CALL AUX(DY1,Y1,T0+HT/3.0,N,LIMIT)
      DO 202 I=1,N
        Y1(I)=Y0(I)+HT/6.0*DY0(I)+HT/6.0*DY1(I)
  202  CONTINUE
      CALL AUX(DY1,Y1,T0+HT/3.0,N,LIMIT)
      DO 203 I=1,N
        Y1(I)=Y0(I)+HT/8.0*DY0(I)+3.0*HT/8.0*DY1(I)
  203  CONTINUE
      CALL AUX(DY2,Y1,T0+HT/2.0,N,LIMIT)
      DO 204 I=1,N
        Y1(I)=Y0(I)+HT/2.0*DY0(I)-3.0*HT/2.0*DY1(I)+
     +        2.0*HT*DY2(I)
  204  CONTINUE
C
C **** AUX SUBROUTINE IS CALLED TO OBTAIN THE ESTIMATES OF
C **** STATE VARIABLES AND Y2 IS EQUIVALENT WITH Y
C
      CALL AUX(DY1,Y1,T0+HT,N,LIMIT)
      DO 205 I=1,N
        Y2(I)=Y0(I)+HT/6.0*DY0(I)+2.0*HT/3.0*DY2(I)+
     +        HT/6.0*DY1(I)
```

```
 205  CONTINUE
      DO 206 I=1,N
C
C **** ERROR TEST FOR ESTIMATES OF STATE VARIABLES Y2(I)
C
      ERROR(I)=DABS(0.2*(Y1(I)-Y2(I)))
      IF(ERROR(I).GT.TOL) THEN
C
C IF ACCURACY IS NOT REACHED THEN STEP LENGTH IS HALVED
C
        HT=HT*0.5D0
C
C IF MATHEMATICAL DISCONTINUITY IS MET, STEP LENGTH COULD BE REDUCED TO ZERO.
C SO LET INTEGRATION PROCESS JUMPS OUT FROM ERROR TEST REGION WHEN HT<HTMIN.
C I.E. INTEGRATION RESULTS Y2(I) ARE CONSIDERED TO BE ACCEPTABLE AT tn+HT POINT.
C
        IF(HT.LT.HTMIN) GOTO 207
        GOTO 102
      END IF
 206  CONTINUE
 207  CONTINUE
C
C **** THE EVALUATION OF STATE VARIABLES AT ONE STEP IS COMPLETED
C **** SET LIMIT=2
C
      LIMIT = 2
C
C **** THE PURPOSE OF CALLING AUX IS TO SOLVE DISCONTINUITIES BY
C **** USING CONDITIONAL STATEMENTS IN MODEL CALCULATION SUBROUTINES
C
      CALL AUX(DY,Y2,T0,N,LIMIT)
C
C **** DERIVATIVES OF STATE VARIABLES ARE STORED INTO TEMPORARY ARRAY
C **** DYTEMP(I)
C
      DO 112 I=1,N
  112   DYTEMP(I)=DY(I)
C
C **** PREPARE FOR ANOTHER STEP
```

```
C
      K=K+1
      IF(B1.OR.B2) THEN
C
C **** IF ACCURACY OF TIME-RANGE ERROR IS SATISFIED, THE STEP IS
C **** COMPLETED AND DO PREPARATION FOR A NEW STEP AND THEN GOTO
C **** LABEL 500 TO OUTPUT RESULTS.
C
        HEST=HOUT
        T=T0+HT
        DO 211 I=1,N
          Y(I)=Y2(I)
  211    CONTINUE
        IDIR=140
        GOTO 500
      END IF
C
      IDIR=102
      T0=T0+HT
      DO 209 I=1,N
        Y0(I)=Y2(I)
  209  CONTINUE
      DO 210 I=1,N
C
C **** IF ERROR IS IN THE TOLERANCE REQUIRED, STEP LENGTH IS UNCHANGED
C
        IF(ERROR(I)*64.D0.GT.TOLM(I)) GOTO 500
  210  CONTINUE
C
C **** OTHERWISE, THE INTEGRATION STEP WILL BE DOUBLED
C
        HT=2.D0*HT
        IF(HT.GT.HTMAX) HT=HTMAX
        GOTO 500
C
  500  CONTINUE
      IF(HT.LT.0.D0) HT=DABS(HT)
C
C ***********************************************************************************************
```

```
C **** BELOW SECTION IS DESIGNED TO OUTPUT INTEGRATION RESULTS ****
C ***************************************************************************************
C
C **** IF CALCULATION TIME IS LESS THAN PRINT TIME, STATE
C **** VARIABLES ARE STORED IN YN(I),THEN GOTO 140
C
      IF(T.LT.TABC) THEN
       HTMAX=TAB
       DO 11 I=1,N
        YN(I)=Y2(I)
 11     IF(DABS(Y2(I)-Y0(I)).GT.TOL) HTMAX=TAB/2.D0
       IF(IDIR.EQ.140) THEN
        GOTO 140
       ELSE
        GOTO 102
       END IF
C
C ***** ELSE, IF CALCULATION TIME EXACTLY EQUAL PRINT TIME
C ***** CALL SUBROUTINE OUT TO OUTPUT SOLUTION (GOTO 21)
C
      ELSE
       DO 15 I = 1,N
         Y(I) = Y2(I)
 15     CONTINUE
       IF(T.EQ.TABC) GOTO 21
C
C ***** OTHERWISE A LINEAR INTERPOLATION PROCESS NOW COMMENCES.
C ***** DLTAY(I) IS NOW USED TO HOLD THE DIFFERENCE BETWEEN THE
C ***** CURRENT TIME STATE VARIABLES AND THE LAST COMPUTED STATE
C ***** VARIABLE VALUES.
C
      HTMAX=TAB
      DO 12 I=1,N
        DLTAY(I)=Y(I)-YN(I)
 12    IF(DABS(Y(I)-Y0(I)).GT.TOL) HTMAX=TAB/2.D0
     END IF
C
C ***** LINEAR INTERPOLATION IS NOW USED TO COMPUTE THE STATE
C ***** VARIABLES AT THE PRINT INTERVAL FROM THOSE AT THE
```

```
C ***** COMPLETED INTEGRATION STEP.
C
      DO 13 I=1,N
 13     YABC(I)=YN(I)+(TABC-(T-HT))*DLTAY(I)/(HT+1.D-20)
C
C ***** THE OUTPUT SUBROUTINE (OUT) IS CALLED TO STORED THE SIMULATION
C ***** RESULTS IN THE RESULTS FILE (CADRES.DAT) AND IF REQUESTED TO
C ***** PRINT OUT THE RESULTS ON THE SCREEN. NORMALLY, THE RESULTS ARE
C ***** PRINTED OUT AT THE USER-DEFINED PRINT INTERVALS.
C ***** ( SET LIMIT = 1 TO OBTAIN THE VALUES OF ALGEBRIC VARIABLES)
C
      CALL OUT(TABC,YABC,N,TAB,1)
C      WRITE(6,'(18HIVARIABLE STEP HT=,2X,1PD20.5)') HT
      GOTO 33
C
C ***** IF THE INTEGRATION TIME (T) IS EQUAL TO THE PRINT TIME (TABC).
C ***** SUBROUTINE OUT IS CALLED AND THE RESULTS ARE PRINTED OUT AT
C ***** A COMPLETED INTEGRATION STEP.
C ***** ( SET LIMIT = 1 TO OBTAIN THE VALUES OF ARGEBRIAC VARIABLES)
C
 21    CALL OUT(T,Y,N,TAB,1)
C      WRITE(6,'(18HIVARIABLE STEP HT=,2X,1PD20.5)') HT
      GOTO 33
 33    CONTINUE
C
C ***** A NEW PRINT TIME (TABC) IS DETERMINED
C
      TABC = TABC + TAB
C
C ***** IF A WHOLE INTEGRATION PROCESS IS NOT FINISHED, GO BACK
C ***** TO LABEL 140
C
      IF(T.LT.TEND) THEN
        IF(IDIR.EQ.140) THEN
          GOTO 140
        ELSE
          GOTO 102
        END IF
      END IF
```

```
C
C **** RETURN TO MAIN AND END OF KUTMER
C
      RETURN
      END
```

LISTING OF KUTFEH INTEGRATION SUBROUTINE

```
C ***************************************************************************************************
C INTEGRATION SUBROUTINE FOR HASP. USES RUNGe-KUTTA-FEHLBERG'S METHOD OF
C ORDER 5. METHOD INTEGRATES A SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
C      OVER A RANGE. HAS PROVISION FOR INTERVAL HALVING
C ***************************************************************************************************
C
C                          SCHOOL OF ENGINEERING
C                          UNIVERSITY OF BATH
C                          CLAVERTON DOWN
C                          BATH
C
C     DEVELOPED AND COMMENTS ADDED:    L.M. WANG
C     DATE:                    12-JAN-86
C
C ***************************************************************************************************
C
C GENERAL COMMENTS ABOUT THE SUBROUTINE:
C
C     (1) THE SUBROUTINE IS CALLED FOR THE FULL DURATION OF A SIMULATION.
C         THE INITIAL SECTION OF THIS SUBROUTINE SETS UP CERTAIN VARIABLES
C         WHICH ARE USED AND CHANGED AS THE SIMULATION PROGRESSES, AND THE
C         MAXIMUM AND MINIMUM STEPSIZE EMPLOYED IN THE SUBROUTINE.
C
C     (2) LABEL 140 IN THE SUBROUTINE IS WHERE THE INTEGRATION STARTS,
C         AND IT IS ALSO THE LABEL WHERE EACH NEW INTEGRATION STEP COMMENCES
C         WHEN AN INTEGRATION STEP HAS COMPLETED SUCCESSFULLY.
C
C     (3) LABEL 120 IS THE LABEL WHERE THE INTEGRATION LOOP COMMENCES
C         WHEN THE INTEGRATION STEPSIZE IS H. LABEL 120 IS ALSO EXECUTED
C         IF STEPSIZE H IS HALVED OR DOUBLED.
C
C     (4) AFTER LABEL 120 AN ERROR TEST IS USED FOR THE CHOICE OF THE
```

```
C       LOGICAL CONTROL VARIABLES B1 AND B2. THE LABELS 101 TO 206 IN
C       THE SUBROUTINE ARE USED TO EVALUATE THE ESTIMATES OF STATE
C       VARIABLES AND THEIR DERIVATIVES.
C
C     (5) IN THE DO LOOP (207) ANOTHER ERROR TEST IS USED TO CONTROL
C       THE STEPSIZE. IF THE ERROR OF ESTIMATES OF STATE VARIABLES
C       IS LARGER THAN A SET TOLERANCE(TOL), THE STEPSIZE IS HALVED
C       AND THEN RETURN BACK TO LABEL 102 TO RE-EVALUATE THE ESTIMATES
C       AGAIN. IF THE INTEGRATION STEPSIZE IS LESS THAN A SET MINIMUM
C       STEPSIZE (HMIN), THE EXECUTION OF FUNCTION EVALUATION JUMPS TO
C       LABEL 208. I.E. THE ESTIMATES OF STATE VARIABLES ARE CONSIDERED
C       TO BE ACCEPTABLE. IN THIS CASE, A POSSIBLE INFINIT LOOP OF ERROR
C       TEST CAN BE AVOIDED.
C
C     (6) IN THE DO LOOP (210) THE ERROR TEST IS EMPLOYED TO DETERMINE
C       WHETHER THE INTEGRATION STEPSIZE NEEDS TO BE DOUBLED.
C
C     (7) A LINEAR INTERPOLATION PROCESS IS USED TO OBTAIN APPROXIMATIONS
C       OF STATE VARIABLES AT EXACT PRINT INTERVAL POINT FROM THOSE AT
C       THE COMPLETED INTEGRATION STEP. THIS PROCESS IS CARRIED OUT FROM
C       DO LOOP 13 TO THE LABEL 33.
C
C ***************************************************************************************************
C
C VARIABLES IN ARGUMENT LIST:
C
C     N     -  NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS TO BE SOLVED
C     HT    -  SIMULATION TIME STEP IN SECONDS
C     T     -  SIMULATION TIME IN SECONDS
C     TEND  -  OVERALL SIMULATION TIME IN SECONDS
C     TAB   -  PRINT INTERVAL IN SECONDS
C     Y     -  ARRAY (OF SIZE N) OF STATE VARIABLES. STORES THE RESULTS
C              IN DATA FILE CADRES.DAT(FOR PLOTTING). USER HAS AN OPTION
C              TO PRINT RESULTS AS SIMULATION PROGRESSES.
C     DY    -  USED TO STORE DERIVATIVES OF FUNCTIONS
C      TOL  -  INTEGRATION TOLERANCE USED FOR ERROR TEST (SET IN SUBROUTINE
C              CONTRL)
C     AUX   -  SUBROUTINE WRITTEN BY THE HASP PROGRAM GENERATOR TO PROVIDE
C              A CALLING LIST FOR THE COMPONENT MODEL SUBROUTINES
```

```
C     OUT   -  SUBROUTINE FOR OUTPUTTING RESULTS OF SIMULATION
C
C ***** VARIABLES USED IN THE INTEGRATION SUBROUTINE
C
C      LIMIT  - LOGICAL CONTROL INDICATOR IN SUBROUTINE
C                    LIMIT = 0 : INITIALISATION PROCESS
C             LIMIT = 1 : INTERMEDIATE CALCULATION PROCESS
C             LIMIT = 2 : WHEN AN INTEGRATION STEP HAS COMPLETED
C                     SUCCESSFULLY.(OR INITIALISATION PROCESS)
C
C       DELTAX -   X-RANGE FOR WHICH THE INTEGRATION IS TO BE DONE IN ONE CALL
C       DLTAY  -   DIFFERENCE BETWEEN CURRENT TIME STATE VARIABLES AND LAST
C              TIME STATE VARIABLES ( Y(I)-YN(I) )
C       DY0,DY1,DY2,DY3,DY4,DY5
C                  -   INTERMEDIATE CALCULATION VALUES OF DERIVATIVES
C       HEST   -   AN ESTIMATED VALUE OF HT WHICH IS PROVIDED TO START OFF
C                    THE SUBROUTINE.
C       HOUT   -   EXCHANGE VARIABLE OF ESTIMATED STEP LENGTH HEST
C       K          -   NUMBER OF INTEGRATION  STEPS TAKEN IN THE RANGE DELTAX
C       T0     -   INITIAL INTEGRATION TIME OF EACH INTERVAL
C      TABC   -   PRINT TIME IN SECONDS
C      Y0,Y1,Y2
C            -   INTERMEDIATE CALCULATION VALUES OF STATE VARIABLES
C      YABC   -   STATE VARIABLES AT PRINT TIME POINT
C      YN     -   LAST COMPLETED STATE VARIABLES
C
C ***********************************************************************************************
C
      SUBROUTINE KUTFEH(N,T,TEND,TAB,Y,DY,TOL,AUX,OUT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /TSTEP/HT
      DIMENSION Y(N),DY(N),Y0(50),Y1(50),Y2(50),DY0(50),DY1(50),
     +          DY2(50),DY3(50),DY4(50),DY5(50),YN(50),YABC(50),
     +          DLTAY(50),ERROR(50),DYTEMP(50)
      LOGICAL B1,B2
C
C **** ASSIGNMENT FOR AN ESTIMATED VALUE OF HT
C
      HEST=HT
```

```
C
C **** ASSIGNMENT FOR MAXIMUM AND MINIMUM STEP LENGTH
C
      HTMAX=1.D30
      HTMIN=1.D-6
C
C **** THE OUTPUT SUBROUTINE IS CALLED TO PRINT OUT INITIAL CONDITIONS
C
      LIMIT=0
      TABC=TAB
      CALL OUT(T,Y,N,TAB,LIMIT)
C
C **** BEGIN TRIAL STEP
C
  140  CONTINUE
C
C **** REASIGN THE VALUES OF T0, K
C
      T0=T
      K=0
C
C **** LET INTEGRATION STEP LENGTH HT BE THE ESTIMATED VALUE OF HT
C
      HT=HEST
C
C **** IF FIRST ENTRY THEN INITIALISE Y0 TO Y
C
      DO 200 I=1,N
        Y0(I)=Y(I)
  200  CONTINUE
C
C **** START INTEGATION LOOP IN ONE TIME-RANGE (DALTAX)
C
  102   B1=.FALSE.
      B2=.FALSE.
      DELTAX=1.5D0*HT
      HOUT=HT
C
C **** ERROR TEST OF DETERMINING LOGICAL VARIABLES
```

```
C
      TEST=(T+DELTAX-T0-HT)/(DELTAX+1.D-25)
      IF(TEST.GT.1.D-6)GOTO 101
      IF(TEST.LT.-1.D-6) THEN
       B1=.TRUE.
       HT=DELTAX-T0+T
       GOTO 101
      END IF
      B2=.TRUE.
C
C **** COMPUTE SIX APPROXIMATIONS TO Y
C **** LET  LIMIT=1
C
  101   LIMIT=1
      IF(HT.GT.HTMAX) HT=HTMAX
      CALL AUX(DY0,Y0,T0,N,LIMIT)
      DO 201 I=1,N
        Y1(I)=Y0(I)+HT/4.0*DY0(I)
  201 CONTINUE
      CALL AUX(DY1,Y1,T0+3.0*HT/8.0,N,LIMIT)
      DO 202 I=1,N
        Y1(I)=Y0(I)+3.0*HT/32.0*DY0(I)+9.0*HT/32.0*DY1(I)
  202   CONTINUE
      CALL AUX(DY2,Y1,T0+12.0*HT/13.0,N,LIMIT)
      DO 203 I=1,N
        Y1(I)=Y0(I)+1932.0*HT/2197.0*DY0(I)-7200.0*HT/2197.0*
     +        DY1(I)+7296.0*HT/2197.0*DY2(I)
  203   CONTINUE
      CALL AUX(DY3,Y1,T0+HT,N,LIMIT)
      DO 204 I=1,N
        Y1(I)=Y0(I)+439.0*HT/216.0*DY0(I)-8.0*HT*DY1(I)+
     +        3680.0*HT*DY2(I)/513.0-845.0*HT*DY3(I)/4104.0
  204   CONTINUE
      CALL AUX(DY4,Y1,T0+HT/2.0,N,LIMIT)
      DO 205 I=1,N
        Y1(I)=Y0(I)-8.0*HT/27.0*DY0(I)+2.0*HT*DY1(I)-
     +        3544.0*HT*DY2(I)/2565.0+1859.0*HT*DY3(I)/4104.0-
     +        11.0*HT*DY4(I)/40.0
  205   CONTINUE
```

```
C
C **** AUX SUBROUTINE IS CALLED TO OBTAIN THE STATE VARIABLES
C **** AND Y2 IS EQUIVALENT WITH Y
C
      CALL AUX(DY5,Y1,T0+HT,N,LIMIT)
      DO 206 I=1,N
        Y2(I)=Y0(I)+(25.0*DY0(I)/216.0+1408.0*DY2(I)/2565.0+
     +        2197.0*DY3(I)/4104.0-DY4(I)/5.0)*HT
  206  CONTINUE
      DO 207 I=1,N
C
C **** ERROR TEST FOR ESTIMATES OF STATE VARIABLES Y2(I)
C

      ERROR(I)=DABS(HT*(DY0(I)/360.0-128.0*DY2(I)/4275.0-
     + 2197.0*DY3(I)/75240.0+DY4(I)/50.0+2.0*DY5(I)/55.0))*0.02D0
      IF(ERROR(I).GT.TOL) THEN
C
C IF ACCURACY IS NOT REACHED THEN STEP LENGTH IS HALVED
C
       HT=HT*0.5D0
C
C IF MATHEMATICAL DISCONTINUITY IS MET, STEP LENGTH COULD BE REDUCED TO ZERO.
C SO LET INTEGRATION PROCESS JUMPS OUT FROM ERROR TEST REGION, IF HT<HTMIN.
C I.E. INTEGRATION RESULTS Y2(I) ARE ACCEPTABLE AT tn+HT POINT.
C
       IF(HT.LT.HTMIN) GOTO 208
       GOTO 102
      END IF
  207  CONTINUE
  208  CONTINUE
C
C **** THE EVALUATION OF STATE VARIABLES AT ONE STEP IS COMPLETED
C **** SET LIMIT=2
C
      LIMIT = 2
C
C **** THE PURPOSE OF CALLING AUX IS TO SOLVE DISCONTINUITIES BY
C **** USING CONDITIONAL STATEMENTS IN MODEL CALCULATION SUBROUTINES
```

```
C
      CALL AUX(DY,Y2,T0,N,LIMIT)
C
C **** DERIVATIVES OF STATE VARIABLES ARE STORED INTO TEMPORARY ARRAY
C **** DYTEMP(I)
C
      DO 112 I=1,N
  112     DYTEMP(I)=DY(I)
C
C **** PREPARE FOR ANOTHER STEP
C
      K=K+1
      IF(B1.OR.B2) THEN
C
C **** IF ACCURACY OF TIME-RANGE ERROR IS SATISFIED, THE STEP IS
C **** COMPLETED AND DO PREPARATION FOR A NEW STEP AND THEN GOTO
C **** LABEL 500 TO OUTPUT RESULTS.
C
        HEST=HOUT
        T=T0+HT
      DO 211 I=1,N
        Y(I)=Y2(I)
  211   CONTINUE
        IDIR=140
        GOTO 500
      END IF
      IDIR=102
      T0=T0+HT
      DO 209 I=1,N
        Y0(I)=Y2(I)
  209   CONTINUE
      DO 210 I=1,N
C
C **** IF ERROR IS IN THE TOLERANCE REQUIRED, STEP LENGTH IS UNCHANGED
C
        IF(ERROR(I)*16.D0.GT.TOL) GOTO 500
  210   CONTINUE
C
C **** OTHERWISE, THE INTEGRATION STEP WILL BE DOUBLED
```

```
C
        HT=2.D0*HT
        IF(HT.GT.HTMAX) HT=HTMAX
        GOTO 500
C
 500  CONTINUE
     IF(HT.LT.0.D0) HT=DABS(HT)
C
C ***************************************************************************************
C **** BELOW SECTION IS DESIGNED TO OUTPUT INTEGRATION RESULTS ****
C ***************************************************************************************
C
C **** IF CALCULATION TIME IS LESS THAN PRINT TIME, STATE
C **** VARIABLES ARE STORED IN YN(I),THEN GOTO 140
C
     IF(T.LT.TABC) THEN
       HTMAX=TAB
       DO 11 I=1,N
         YN(I)=Y2(I)
 11      IF(DABS(Y2(I)-Y0(I)).GT.TOL) HTMAX=TAB/2.D0
       IF(IDIR.EQ.140) THEN
         GOTO 140
       ELSE
         GOTO 102
       END IF
C
C ***** ELSE, IF CALCULATION TIME EXACTLY EQUAL PRINT TIME
C ***** CALL SUBROUTINE OUT TO OUTPUT SOLUTION (GOTO 21)
C
     ELSE
       DO 15 I = 1,N
         Y(I) = Y2(I)
 15      CONTINUE
       IF(T.EQ.TABC) GOTO 21
C
C ***** OTHERWISE A LINEAR INTERPOLATION PROCESS NOW COMMENCES.
C ***** DLTAY(I) IS NOW USED TO HOLD THE DIFFERENCE BETWEEN THE
C ***** CURRENT TIME STATE VARIABLES AND THE LAST COMPUTED STATE
C ***** VARIABLE VALUES.
```

```
C
      HTMAX=TAB
      DO 12 I=1,N
        DLTAY(I)=Y(I)-YN(I)
 12   IF(DABS(Y(I)-Y0(I)).GT.TOL) HTMAX=TAB/2.D0
      END IF
C
C ***** LINEAR INTERPOLATION IS NOW USED TO COMPUTE THE STATE
C ***** VARIABLES AT THE PRINT INTERVAL FROM THOSE AT THE
C ***** COMPLETED INTEGRATION STEP.
C
      DO 13 I=1,N
 13     YABC(I)=YN(I)+(TABC-(T-HT))*DLTAY(I)/(HT+1.D-20)
C
C ***** THE OUTPUT SUBROUTINE (OUT) IS CALLED TO STORED THE SIMULATION
C ***** RESULTS IN THE RESULTS FILE (CADRES.DAT) AND IF REQUESTED TO
C ***** PRINT OUT THE RESULTS ON THE SCREEN. NORMALLY, THE RESULTS ARE
C ***** PRINTED OUT AT THE USER-DEFINED PRINT INTERVALS.
C ***** ( SET LIMIT = 1 TO OBTAIN THE VALUES OF ALGEBRIC VARIABLES)
C
      CALL OUT(TABC,YABC,N,TAB,1)
C       WRITE(6,'(18HIVARIABLE STEP HT=,2X,1PD20.5)') HT
      GOTO 33
C
C ***** IF THE INTEGRATION TIME (T) IS EQUAL TO THE PRINT TIME (TABC).
C ***** SUBROUTINE OUT IS CALLED AND THE RESULTS ARE PRINTED OUT AT
C ***** A COMPLETED INTEGRATION STEP.
C ***** ( SET LIMIT = 1 TO OBTAIN THE VALUES OF ARGEBRIAC VARIABLES)
C
 21   CALL OUT(T,Y,N,TAB,1)
C       WRITE(6,'(18HIVARIABLE STEP HT=,2X,1PD20.5)') HT
      GOTO 33
 33   CONTINUE
C
C ***** A NEW PRINT TIME (TABC) IS DETERMINED
C
      TABC = TABC + TAB
C
C ***** A WHOLE INTEGRATION PROCESS IS NOT FINISHED, GO BACK TO LABEL 140
```

```
C
      IF(T.LT.TEND) THEN
       IF(IDIR.EQ.140) THEN
         GOTO 140
       ELSE
         GOTO 102
       END IF
      END IF
C
C **** RETURN TO MAIN
C
      RETURN
C
C ***** END OF KUTFEH
C
      END
```